



PRESS

TYP03

Werner Altmann
René Fritz
Daniel Hinderink



EYROLLES

TYPO3

Werner Altmann
René Fritz
Daniel Hinderink

EYROLLES



ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Traduction autorisée de l'ouvrage en langue allemande intitulé :
TYPO3 – Enterprise Content Management (ISBN : 3-937514-01-5)
de Werner Altmann, René Fritz et Daniel Hinderink
© 2004, Open Source Press, Munich (Allemagne)
<http://www.opensourcepress.de>

Adapté de l'allemand par Nicolas Wezel de la société Streamsys.



Le code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée notamment dans les établissements d'enseignement, provoquant une baisse brutale des achats de livres, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre Français d'Exploitation du Droit de Copie, 20, rue des Grands-Augustins, 75006 Paris.

© Open Source Press, Munich, 2004, pour l'édition originale en langue allemande

© Groupe Eyrolles, 2006, pour la présente édition, ISBN : 2-212-11781-7

Table des matières

Préface de Kasper Skårhøj	11
Préface des auteurs	17
À propos de ce livre	19
Introduction	21
1 Introduction	23
1.1 Historique	23
1.2 Qu'est-ce qu'un CMS?	24
1.2.1 Séparation du contenu de la forme	25
1.2.2 Cycle de vie du contenu	25
1.2.3 Système modulaire	26
1.2.4 Groupe cible	26
1.2.5 Références TYPO3	26
1.3 La communauté TYPO3	27
1.4 L'association TYPO3	29
1.4.1 Objectifs	29
1.4.2 Être membre et s'inscrire	29
2 Installation	31
2.1 Choix du paquetage TYPO3 et de la configuration du serveur	31
2.1.1 Matériel	32
2.1.2 Serveur Web	35
2.1.3 Base de données	35
2.1.4 Autres logiciels	36
2.1.5 Choix du paquetage TYPO3	36

2.2	Installation de test et d'initiation	38
2.2.1	Installation WAMP sous Windows	38
2.2.2	Installation Quick Install sous Linux	39
2.3	L'installation en production	40
2.3.1	Installation LAMP	40
2.3.2	Installation WAMP	43
2.3.3	Installation WIIS	44
2.4	L'outil d'installation	45
2.4.1	Basic Configuration	46
2.4.2	Database Analyzer	47
2.4.3	Image Processing	49
2.4.4	All Configuration	49
2.4.5	typo3temp/	49
2.4.6	phpinfo()	49
2.4.7	Edit files in typo3conf/	49
2.4.8	About	50
2.5	Options de configuration dans TYPO3_CONF_VARS	50
2.5.1	[GFX]:\$TYPO3_CONF_VARS["GFX"]	50
2.5.2	[SYS]:\$TYPO3_CONF_VARS["SYS"]	53
2.5.3	[EXT]:\$TYPO3_CONF_VARS["EXT"]	58
2.5.4	[BE]:\$TYPO3_CONF_VARS["BE"]	59
2.5.5	[FE]:\$TYPO3_CONF_VARS["FE"]	64
2.5.6	Autres options	70
2.6	Séparation du serveur de production et du serveur en ligne	70
2.6.1	Pages statiques	71
2.7	Sauvegardes	71
2.8	Mises à jour	73
2.9	En cas de problème	74

TYPO3 pour les rédacteurs **75**

3	TYPO3 pour les rédacteurs	77
3.1	Le rôle du rédacteur	78
3.2	Accéder au système	79
3.2.1	Configuration du navigateur	79
3.2.2	Identification	79

3.3	Interface utilisateur et modules	80
3.3.1	Zones de l'interface utilisateur	80
3.3.2	Modules principaux et sous-modules – un aperçu	83
3.4	Le module utilisateur → centre de tâches comme centre de communication	87
3.5	Les pages, réceptacles de contenu	91
3.5.1	Structure d'un site, arborescence et éléments de contenu	91
3.5.2	Créer et éditer de nouvelles pages	92
3.5.3	Différents types de pages	95
3.6	Insertion d'éléments de contenu dans TYPO3	99
3.6.1	Création et édition de nouveaux éléments de contenu	101
3.6.2	Types de contenu	103
3.7	Ressources dans TYPO3	115
3.7.1	Gestion des ressources dans l'arborescence des fichiers	115
3.7.2	Insérer des ressources dans une application	116
3.8	Édition frontend	117
3.9	Le Rich Text Editor	119
3.10	Travailler efficacement avec TYPO3	123
3.10.1	Scénario	124
3.10.2	Créer l'arborescence des pages	124
3.10.3	Presse-papiers	126
3.10.4	Éditer des champs sélectionnés	127
3.10.5	Raccourcis	129
3.10.6	Aide au niveau du contenu	130
3.10.7	Restaurer/éditer l'historique	132
3.10.8	Multilinguisme	134

TYPO3 pour les administrateurs 137

4	TYPO3 pour les administrateurs	139
4.1	Tâches et objectifs de l'administration	139
4.2	Planifier et installer l'environnement de gestion de contenu	141
4.3	Principes d'organisation des droits d'accès dans TYPO3	142
4.3.1	Exemple pratique	143
4.3.2	Étapes de mise en œuvre	144
4.4	Administration des utilisateurs backend	144
4.4.1	Créer des groupes d'utilisateurs	145

4.4.2	Créer des comptes utilisateurs	150
4.5	Administration des utilisateurs à l'aide du module Outils → Administration des utilisateurs	152
4.6	Droits d'accès au niveau de la page	153
4.7	Édition frontend pour utilisateurs backend	155
4.8	TSconfig – options et interface	157
4.8.1	Assistant TSConfig : consulter les propriétés TypeScript	157
4.8.2	TSConfig utilisateur	157
4.8.3	TSConfig page	159
4.8.4	Ajustement du Rich Text Editor	162
4.8.5	Le module Web → Info → Configuration TS de la page	167
4.9	Créer des Workflows simples	167
4.9.1	Configuration d'un workflow	168
4.9.2	Exemple : workflow d'Actualités	168
4.10	Procédures et actions	170
4.10.1	Types d'actions	171
4.10.2	Exemple : action pour créer des utilisateurs	172
4.11	Administration des utilisateurs frontend	175
4.11.1	Création de groupes d'utilisateurs	175
4.11.2	Création de comptes utilisateurs	175
4.11.3	Identification	176
4.11.4	Assigner des pages et des éléments de contenu	176
4.11.5	Perspectives	177
4.12	Statistiques et logs	177
4.12.1	Le module Web → Info	178
4.12.2	Intégration d'AWStats	178
4.12.3	Analyse des fichiers journaux	179
4.12.4	Logs frontend	180
4.12.5	Le module Vérification BD	180
4.13	TYPO3 et le système de cache	183
4.14	Digital Asset Management	184
4.14.1	Tâches et buts du DAM	184
4.14.2	Intégration dans TYPO3	185
4.14.3	Perspectives	190
4.15	Administration : l'avenir	191

TYPO3 pour les développeurs	193
5 TypoScript	195
5.1 Le rôle du développeur	195
5.1.1 Le processus de mise en œuvre	195
5.1.2 Prérequis et vue d'ensemble	196
5.2 TypoScript — Principes de base	197
5.2.1 Qu'est-ce que TypoScript ?	197
5.2.2 TSref	199
5.2.3 Digression : TypoScript et PHP	199
5.2.4 Gabarits TypoScript	201
5.2.5 Hello World! — Le premier gabarit TypoScript	203
5.2.6 Cascade de gabarits	205
5.2.7 Enregistrements de gabarits	206
5.2.8 Constants et Setup	211
5.2.9 Éléments et concepts	212
5.2.10 La syntaxe	214
5.2.11 Ordre de traitement	222
5.2.12 L'emboîtement d'objets	223
5.3 Objets, fonctions et types de données TS	225
5.3.1 Types de données	225
5.3.2 Le concept d'enveloppe	227
5.3.3 Fonctions	227
5.3.4 Objets de contenu (cObjects)	229
5.3.5 Objets de premier niveau	237
5.4 Outils de développement	241
5.4.1 Info/Modify	241
5.4.2 Assistant TS	242
5.4.3 TypoScript Object Browser	242
5.4.4 Template Analyzer	244
5.4.5 Constant Editor	245
5.4.6 Panneau d'Administration	251
5.4.7 Import et export de pages TYPO3	252
5.5 Gabarits standards (gabarits statiques)	255
5.5.1 content (default)	257
5.5.2 styles.*	257

5.5.3	cSet.*	258
5.5.4	frameset;*	259
5.5.5	template;*	259
5.5.6	plugin.*	262
5.5.7	temp.*	263
5.5.8	content.tt_*	263
5.5.9	(example)	264
5.5.10	language.*	264
5.6	Les bases de la mise en page — Concepts de gabarit	264
5.6.1	Gabarits standards (gabarits statiques)	264
5.6.2	Gabarits TypoScript purs	265
5.6.3	Gabarits TypoScript et HTML	265
5.6.4	Template Auto-Parser	266
5.6.5	TemplaVoilà	266
5.7	Restitution du contenu	267
5.8	Changer de gabarits avec type/typeNum	268
5.9	Création de gabarits TypoScript	269
5.9.1	TypoScript et gabarits HTML	272
5.9.2	Le Template Auto-Parser	284
5.9.3	Gabarits TypoScript purs	290
5.10	Menus	296
5.10.1	Le cObject HMENU — propriétés générales des menus	298
5.10.2	Menus de texte (TMENU)	300
5.10.3	Menus graphiques (GMENU)	304
5.10.4	Menus basés sur des couches (TMENU_LAYERS/ GMENU_LAYERS)	308
5.10.5	GMENU_FOLDOUT	311
5.10.6	ImageMaps (IMGMENU)	314
5.10.7	Menus JavaScript (JSMENU)	317
5.10.8	Menus .special	318
5.11	TypoScript en détail	322
5.11.1	La fonction optionSplit	322
5.11.2	Travailler avec des images et le GIFBUILDER	325
5.11.3	La fonction stdWrap	331
5.11.4	Conditions	339
5.12	Travailler avec des cadres	345

5.12.1	Création de cadres	346
5.12.2	Le site exemple avec des cadres	348
5.13	Futur et perspectives	353
5.13.1	XHTML et accessibilité	353
5.13.2	Accessibilité	356
5.13.3	TemplaVoilà	358
6	Extensions	367
6.1	Aperçu	367
6.2	Le système d'extensions	368
6.2.1	Structure d'extensions	368
6.2.2	Clé d'extension	369
6.2.3	Composants d'extensions	369
6.2.4	Catégories d'extensions	370
6.2.5	Installation: niveau système, global ou local	371
6.2.6	Répertoire d'extensions	371
6.2.7	Documentation	373
6.3	Gestionnaire d'extensions	373
6.3.1	Liste des extensions disponibles	374
6.3.2	Importer des extensions du répertoire	375
6.3.3	Le Kickstarter	377
7	Développement d'extensions	379
7.1	Un compteur de visiteurs en 20 minutes	380
7.2	Assistant d'extensions : le Kickstarter	387
7.2.1	Définition d'une clé d'extension	388
7.2.2	Composants de Kickstarter	388
7.2.3	Structure d'une extension	390
7.2.4	Règles de base des extensions	392
7.3	Gestion d'extensions pour les programmeurs	393
7.3.1	Fonctions du gestionnaire d'extensions	393
7.3.2	Compte utilisateur TER	395
7.3.3	Transfert d'une extension vers le TER	396
7.3.4	Gestion d'extensions TER	397
7.3.5	Publication de documentation	398
7.4	Le framework TYPO3	400
7.4.1	Structure du framework	400

7.4.2	Conventions d'écriture	401
7.4.3	Structure des répertoires	402
7.4.4	Bibliothèques	406
7.4.5	L'API d'extension	409
7.4.6	Structure de base de données	410
7.4.7	Base de données, TCA et TCEForms	414
7.4.8	Flexforms	419
7.4.9	<i>TYPO3 Core Engine</i> (TCE)	422
7.4.10	SQL et tables définies dans le TCA	425
7.4.11	Utilisateurs, sessions et identification	426
7.4.12	Programmation TYPO3 et plate-forme	429
7.4.13	Multilinguisme	432
7.4.14	Codage des caractères	433
7.5	Programmer dans le frontend : les principes	438
7.5.1	Frontend : restitution du contenu	438
7.5.2	API frontend	439
7.5.3	TypoScript frontend (TSFE)	440
7.5.4	cObject, tslib_cObj	441
7.5.5	Restitution des cObjects par PHP	444
7.5.6	tslib_pibase	445
7.5.7	Liens et paramètres dans les plugins	446
7.5.8	USER, USER_INT, cache et paramètres	448
7.6	Programmation frontend : exemples	453
7.6.1	Bordures d'éléments de contenu	453
7.6.2	La balise Typo de compte à rebours (TypoTag)	457
7.6.3	Balise de compte à rebours en JavaScript	461
7.6.4	Intégration de scripts PHP externes	465
7.6.5	Portage de script PHP	470
7.7	Programmation du backend : principes	476
7.7.1	Structure d'un module	476
7.7.2	Module: framework	478
7.7.3	Modules : script	480
7.7.4	Module principal	481
7.7.5	Fonctions de sous-modules	482
7.8	Programmation backend : exemple	482
7.8.1	Outils → Dernières modifications	482

7.8.2	Fonction de sous-module Web → Fonctions → Assistants	494
7.8.3	Menu contextuel	504
7.8.4	Habillages – Changer l'apparence du backend	507
7.9	Services	510
7.9.1	Mise en œuvre des services	510
7.9.2	Développer des services	512
7.9.3	Configuration	518
7.9.4	Introduction d'un nouveau type de service	519
7.10	XCLASS : modification et extension de classe	519
7.11	TYPO3 et autres langages de programmation	521
7.12	Outils pour le développeur	521
7.12.1	ExtDevEval	521
7.12.2	Débogage avec <code>debug()</code>	522
7.12.3	Débogage avec <code>t3lib_div::devLog()</code>	524
7.12.4	FE Debug/Info output et BE Env-Info	526
7.12.5	Environnements de développement PHP	527
Index		529

Préface de Kasper Skårhøj

There is a picture of a child standing on the shelf next to me here in my office. It was in my grandmother's possession until she moved to an old people's home. The boy in the image holds an object in his hand while looking at the camera, interrupted from his investigations by the photographer. I don't have kids yet myself so ... the boy is me as a three-year old.

The picture fascinates me because it helps me to understand myself and God's design of our creative souls. It rips me out of TypoScript, PHP variables and foreach-loops for a moment and puts a smile on my face. The picture captures the essence of my personality, which has always been curious and creative; from building playhouses on my uncle's farm, shooting VHS-movies as a teenager, wiring up a house automation system to, well, programming a "little" CMS tool which I accidentally needed in the early days of the passion we all share; the modern Internet.

Opening the playground

Creativity defines a lot of who I am. TYPO3 has been the output valve of this energy. It is the "Very best of" album from my life until now, even with all the quirks it has. I love TYPO3 because I know it is an authentic expression of my creative pulse. But how did it ever come this far?

Let's turn back the clock to 1997 when I began to study at the Technical University of Denmark. I think after 5 minutes at the introduction course I had spotted another restless soul in the class for whom Taylor polynomials had no significance for the greater meaning in life. We teamed up, I created my own little company (Curby Soft Multimedia) and college was where I slept the extra hours I missed at night. Back then I was extremely inspired by David Siegel's visions for webdesign (www.killersites.com). In fact my "creativity valve" pointed in the visual direction back then; my team mate handled the Linux-stuff, I just juggled around with the colors.

The binary brainwash

The CMS mantra of "separating content from code" was not the brilliant idea of any one individual, but what happened to all of us at that time; everyone realized that customers needed a CMS tool to maintain their websites. In late 1997 we began the first prototypes of what some years later became TYPO3. Unfortunately my team mate didn't deliver the technical work and my proactive gene autoresponded by closing down Photoshop and starting up Homesite, a webbrowser pointed at www.php.net and the installation of a RedHat 6.2

server; one week later I had the minimum knowledge required to make lookups in MySQL and present the content in an HTML page. And I hated it.

At the time in 1998 I met another guy with a lot more marketing experience than myself. He wanted to commercialize the early version of TYPO3. So we set up the "Superfish.com" company together, hired employees and even managed to team up with Saatchi & Saatchi in Copenhagen. But I had one condition; that I would be relieved from my role as the programmer behind the CMS as fast as possible! It was "Too much work and no fun makes Jack a dull boy" for me and my inside was a desert. I was crying out for colors, poetry and visual universes to explore. I hated programming, it dried me out and I had accepted it only as a temporary necessity.

Maybe this should scare me, but guess what; now the pain of programming is gone! Slowly I was swallowed by PHP until the toxic influence of booleans and arrays made me forget my visual gold age; the "re-coding" of a designer into a programmer was successfully complete. But even now, the spring of creativity couldn't be suppressed. Rather than being expressed in visual terms it permeated the making of TYPO3; programming is an art! Programming contains love and passion for beauty just as much as photography or oilpainting does.

Goodbye World, Hello GPL

When I realized that running a company like Superfish was not my cup of tea, nothing mattered more than my creation, TYPO3. I left the company with the rights to the code, my partner kept the rest. I felt I had passed some kind of "Point of no Return" and all I wanted was to finish my work. For what purpose? That question wasn't even asked at the time. I agreed with myself to do just enough freelance work to live and put the rest of my time into finishing TYPO3 over the next six months (it's funny to look back at the repeatedly naive time-estimates I have made in the past – and still do :-).

During my short flirtation with the commercial production of TYPO3, I realized that commercial pressures tend to corrupt the quality of the product; ready or not, it has to go on the shelf to create income. That was one thing I couldn't accept as an artist; I believed in quality, and compromise in this area was no option to me. On the other hand, with no company behind I could probably sell my CMS solution to only a handful of local companies. Compared to the perspectives of sharing TYPO3 with a whole world and thereby helping thousands of people, the latter would far outweigh the joy of making a bit of money locally in Denmark – even if I never saw a single Euro coming back. So I chose the GPL way.

Living waters

Giving away TYPO3 for free also has a strong root in my faith in Jesus. I have been raised in a Christian family and always believed in God. But passionate faith can't be inherited and my most recent "conversion" happened after having worked 16 hours a day for a long time. I think many of you reading this book know the situation and how passion can drive you into intense work. It can be fun and rewarding. But in the long run it dries you out inside, and eventually you ask yourself: what is the point, what am I living for? This is where some people burn out and get depressed. For me it triggered the logical question; If I really believe in the Bible, why not open it and read about what a balanced life should be like?

This had a dramatic impact on my life. I began to take my faith seriously and re-align my actions with my beliefs. I met my wife, Rie. And I discovered my identity and some personal gifts, so I could understand my "mission" in life. I also thought about how you could live in a Christian way in the modern world. I read "feed the hungry" and found that a useful tool like TYPO3 would be just that. I read "love your neighbour" and found that sharing TYPO3 as the best I had would be an act of love. I read "seek and you shall find" and thought that TYPO3 could be just such a reward for those whose who dare to search for alternatives. I read "you got it for nothing, give it for nothing" and thought that TYPO3 was possible through a talent I could only attribute to God, so what would be lost by giving it away?

Reasons to believe

Mixing TYPO3 with Jesus must seem strange to most of you. Why would an apparently intelligent guy believe in something fuzzy like God? But in fact I think programmers have nothing but good reasons to conclude that some kind of higher intelligence must exist. Every day I spend hours writing characters carefully combined into a computer program. I know that a single misplaced byte will make TYPO3 fail to run. I also know that sometimes I need to make an internal redesign which does not add new functionality to TYPO3, but merely opens the possibility of further development. And who would believe me if I claimed to have created TYPO3 by repeatedly combining random bytes and trying to execute them until something useful came out – even if doing this a trillion times? TYPO3 required conscious, intelligent design! I have to admit that the complexity of life points to something outside the universe itself. It doesn't put a name tag on who is behind it all, but there are good reasons for believing in a mastermind.

The 24/7 lane

I declare that I am not a perfectionist. Perfection is not obtainable for humans, it is an ideal. And although the ideal of perfection is our beacon of light, we have to settle for less. What is obtainable is completeness. Completing what you have started is what gives the first step you took a meaning.

TYPO3 is my baby, it takes enormous amounts of my time and often it consumes most of my awareness. When people ask me about TYPO3 and my own working life, I ask them to think about how it was preparing for exams or writing a large thesis at university. I believe that captures the intensity of how my life has been for the last four years, while TYPO3 has been a public project.

It has been a privilege, fun, challenging all along but it has also worn me out. The most precise way to describe this state is to compare me with a fragile ecosystem. Even small changes in the environment can have great impacts on stability. Luckily I have developed an equally good understanding of my inner self and daily I try to walk the roads that motivate, rather than those which lead to despair. I have learned to focus on single issues, and suppress the view from the top of the mountain, which can be overwhelming. I have had to trust myself to be right many times, when it would have been fair to have doubts. And I have learned to strive for perfection but settle for something complete, and sometimes less. It's a strategy of survival, and without it the wave you are surfing will swallow you.

Growing a community

It's easy for me to remember for how long I have been married to Rie; I just think about when TYPO3 was first released to the public – that was also in August 2000 :-). Anyhow, the launch of TYPO3 to the public was supposed to be the end of the line but it became a whole new beginning! At the time I was exhausted after developing for a year on my own with no external response. I remember how lonely I felt. Publishing TYPO3 under GPL changed all this and the growing community became a solid source of power that changed the whole perspective of the project; suddenly my work mattered to someone! This was a fulfillment of my personal "prophecy" that giving TYPO3 away for free would be much more valuable than selling licenses to local customers in Denmark. In addition the new situation greatly compensated for the loneliness in the office, since I now had virtual colleagues all around the globe!

As an Open Source project, a small community quickly grew up around TYPO3, including René Fritz (co-author of this book) as one of the very first personalities on the scene. People contributed by setting up mailing lists, archives, providing support to others, creating small plugins, offering help to port TYPO3 to Windows and most significantly, translating TYPO3 into their native languages.

In April 2001 Rie and I conceived the idea of arranging a snowboard tour for the community and the next winter we did it! 25 people showed up and suddenly faces were attached to email addresses. The first snowboard tour was an amazing event and Jan-Hendrik Heuing would still quote me for saying "I'm starting to believe in it" back then. The year after we were 50 people snowboarding the slopes of Splügen, the next year we were more than 80 gathered in Kitzbühel. More than anything else, the annual snowboard event has become the identity of the TYPO3 community.

Life in The Bazaar

The community of TYPO3 has grown at an exponential rate ever since. From being a small village where everyone knew each other on the mailing lists, it is now a big city with all that that entails. The manpower to help is far greater but the risk of getting impersonal is equally high. I often receive emails from people asking me support questions. I have to delete them flatly. Even answering back that they should use the mailing list can become stressful to me and it really breaks my heart, because on the other hand I hold the ideal to be personal to everyone. But today I have to trust that the community will take good care of the newbie asking for guidance, while I optimize my time for general development which helps thousands, rather than a single person.

The growth of the community also holds great developmental potential. Centered around code contributions via the Extension Repository, it is directly possible for anyone to contribute quality code to the system in a safe way, which protects the integrity of everyone's work, as well as their motivation for contributing.

My greatest vision for TYPO3 is extensions. I strongly believe they are the most perfect vehicle for bringing broad innovation to the project and offering maximum freedom for every developer to demonstrate his or her personal love for the art and beauty of coding web applications. This is the democracy of our community, everyone has equal chances.

The challenge we face, as I see it, is to maintain the friendly atmosphere for which we have traditionally been known. We also have to maintain an effective framework for contributions from the growing number of code authors and apply more quality assurance to contributions of all kinds. In another field we have realized the need to enforce the GPL license, as TYPO3 has become a popular software which obviously would be nice to re-brand and sell as one's own work. And finally, we have to fight the prejudiced minds that think Open Source has nothing to offer, since there is no fee to pay.

Credits

So many people truly deserve to be mentioned here. Unfortunately any attempt to list some names would exclude others equally merited. It is like inviting people to your wedding; It's not hard to invite your best friend, but it is hard to find the criteria to decide who you will exclude, since there is not room for everyone.

However it will be safe to mention my wife, Rie. She is my best friend, she loves me and challenges me. She prays for me and we share faith in God. She has followed TYPO3 all the way and supported every bit of it, often with personal sacrifice when I was stressed out and mentally absent. She has accepted that she is second choice at times, and we all owe her big time for that.

I want to mention Christian Jul Jensen (Denmark) who has been my good friend and mental support through the years, and also my right hand in professional matters. Christian has been my personal proxy for a while, taking the load from my shoulders as times changed and TYPO3 needed more of my dedication, rather than me helping old customers. His help has been priceless.

Daniel Hinderink is another cornerstone in the history of TYPO3. Daniel is not only professional and very skilled. In addition to this he has been a showcase of proactivity to me. Daniel has taken the initiative and become the solution to problems, rather than a part of them. As the coordinator of marketing efforts for TYPO3, he carries a lot of the responsibility for the marketing success TYPO3 has had. But his initiatives have borne even more fruit in areas such as initiating innovation, team building and project organisation. I'm impressed and thankful!

Now the list of names would explode if I wanted to thank everyone who has contributed to TYPO3 with code, support or has otherwise been active in the community. Money is good but sharing your talents in the community is worth even more! Thank you so much everyone. I hope you can recognize the value of sharing the best you have got with the world, as my experience described in this preface has been, and I encourage you hang on in there!

I have received an increasing amount of money donations during the time TYPO3 has been public. To everyone who has sent money I also want to say thanks from my heart. You have enabled me to spend more and more time on TYPO3 rather than doing irrelevant freelance work. Your donations prove how many small streams make one large flood which eventually can power an Open Source project into stable and continual development. I encourage you to stay true to your promises of financial support so we can employ more people developing TYPO3 in the future!

I also want to mention Dassault Systèmes web department in Paris for their generosity and the inspiring friendship we have shared during my times in Paris. Through their belief in TYPO3 they have supported the development in adventurous ways.

Finally I want to say "Hello" to all my future friends in the community! TYPO3 and the snowboard tours are a social pit-stop for me. These relations somehow make the hours behind the screen less lonely and in some cases sparks real-world friendships like the one I have been so lucky to establish with Robert Lemke from Lüneburg.

Three men in blue overalls

To me the third snowboard tour in 2004 was a fantastic experience. I saw old friends again, I got my own snowboard this time, and I met a lot of new, inspiring people. And finally it demonstrated the powerful initiative of the "self-ignited fireworks" that has popped up in the community. The men in the mirror finally jumped into their blue overalls and began work.

Even though the history of TYPO3 is more than I could ever ask for, I always joked about the day when there would be a book about TYPO3 on the shelves in my local bookstore. Three talented community members have now made this dream come true. They asked for a book and found the answer to its creation in their own mirror reflections. I'm thrilled about the outcome and thankful for your contribution to the big picture of completeness we are striving for.

Enjoy the book and welcome to the TYPO3 corner of cyberspace!

– kasper

Préface des auteurs

TYPO3 est un système de gestion de contenu Open Source qui rencontre un immense succès et est réputé être très puissant, mais aussi fort complexe. Dans ce livre, nous avons essayé de décrire TYPO3 dans les grandes lignes, fournissant ainsi un aperçu de son utilisation tant pour les rédacteurs que pour les administrateurs et les développeurs.

En raison des nombreuses références et de la quantité de didacticiels existants, nous avons soulevé la question de l'apport potentiel de ce livre. Le flot de questions des listes email n'a apparemment pas été endigué par la documentation existante. Après une longue observation des problèmes et des questions les plus caractéristiques, nous avons décidé d'écrire un livre s'efforçant d'illustrer les principes de TYPO3 et de démontrer ses applications pratiques à travers des exemples. Ce livre ne doit donc pas être considéré comme un substitut aux références et aux didacticiels de typo3.org ; il devrait plutôt faire office de lien servant à développer une image cohérente de ce qu'est TYPO3 pour les débutants, les utilisateurs et les développeurs, en leur permettant d'y naviguer par eux-mêmes.

Le projet TYPO3 a tâché de diviser tous les niveaux de documentation et de communication en trois groupes, afin de simplifier la navigation. Ce livre procédera de même :

1. Rédaction : décrit les outils du système pour la création du contenu et décrit les méthodes en vue de leur utilisation pratique.
2. Administration : inclut les tâches d'organisation nécessaires à l'implémentation de la gestion de contenu en utilisant TYPO3.
3. Développement : décrit la création d'un site Web et de son interface graphique en utilisant les gabarits, ainsi que la programmation de vos propres applications dans la structure de TYPO3.

Pour les responsables qui ne sont pas encore familiarisés avec la Gestion de Contenu, et avec la Gestion de Contenu en Entreprise en particulier, nous proposons une introduction au début de ce livre ; tout au long de l'ouvrage, nous revenons régulièrement aux points importants en rapport avec les divers domaines des tâches mises en place au sein de l'entreprise.

Les chapitres 1 et 2 introduisent les bases théoriques de la gestion de contenu. Dans ce contexte, les avantages fournis par TYPO3 deviennent évidents, et, à partir de là, nous établissons des bases afin de vous permettre de prendre des décisions concernant son utilisation stratégique. Par ailleurs, ces chapitres introductifs donnent aux lecteurs dépourvus de connaissances antérieures sur le sujet une description de celui-ci, en présentant les termes

et les concepts les plus importants. Cet exposé est suivi du détail de l'installation et de la configuration de TYPO3.

Dans le chapitre 3, nous faisons une démonstration du fonctionnement du système en nous servant de situations concrètes de production de contenu. Un outil complexe doit prouver sa valeur d'une certaine façon, grâce à la facilité de prise en charge de son interface. Après avoir présenté les options de TYPO3 et leurs fonctions, la section se clôt sur un exemple pratique de travail effectif avec TYPO3.

Le chapitre 4 couvre l'administration du système, puis les adaptations aux conditions et aux procédés définis par les producteurs dans leur travail avec le système. En procédant de la sorte, nous montrons par des exemples comment les moyens disponibles s'imbriquent, et comment ils sont utilisés en pratique.

Le chapitre 5 décrit la production de sites Web grâce à l'utilisation de TYPO3. Commençant par l'installation, nous discutons ensuite de la programmation des gabarits avec TYPO3 et passons en revue les différentes méthodes disponibles.

Dans les chapitres 6 et 7, nous présentons l'interface d'extension de TYPO3, le Système Extensions, décrivant les bases et les façons de développer vos propres extensions fonctionnelles dans la structure de TYPO3. Ici, nous pouvons observer, du point de vue du développeur, l'intégration avec les fonctions du noyau et avec les différentes parties de l'architecture de TYPO3 qui peuvent être étendues.

Le texte tout entier comporte des notes de bas de page et ce que nous appelons des « références ». Les notes de bas de page sont destinées à encourager la lecture et l'approfondissement de domaines qui ne relèvent pas directement des aspects techniques de TYPO3. Les références servent de lien entre le livre et typo3.org, ainsi que d'autres ressources.

En saisissant le code numérique sur le site [typo3.org](http://www.typo3.org/book/) (<http://www.typo3.org/book/>), vous serez amené au sujet qui y correspond sur la documentation en ligne, ou bien à des sources plus détaillées. De cette manière, les références techniques et la documentation sont incluses et sont mises à jour autant que possible. Le lecteur a également l'opportunité de découvrir les ressources thématiques et structurelles en ligne, qui lui donnent ainsi un certain sens de l'orientation dans la profusion toujours grandissante d'informations.

À propos de ce livre

Ce dont vous avez besoin pour utiliser ce livre

Il vous sera nécessaire d'avoir installé TYPO3 sur un serveur Web sur lequel sont déployés PHP4 et MySQL. Vous pourriez aussi avoir besoin d'options supplémentaires telles que ImageMagick, GDLib/Freetype, zlib et un accélérateur PHP comme Zend.

Conventions

Dans cet ouvrage, vous trouverez divers styles de textes servant à distinguer différentes sortes d'informations. Voici quelques exemples, ainsi qu'une explication de leur signification.

Il existe trois styles pour les lignes de code. Les mots du code sont indiqués comme suit dans le texte : « si nous souhaitons transformer la couleur du fond d'écran en un joli gris, nous pouvons aller dans `typo3_styles` et personnaliser `base_properties` ».

Si nous rencontrons un bloc de code, il apparaîtra comme suit :

```
$result = $this->query($query);  
$row = $this->fetch_array($result);  
  
$result = $GLOBALS['TYPO3_DB']->sql_query($query);  
$row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($result);
```

Lorsque nous voulons attirer votre attention sur une partie d'un bloc de code, les lignes en question seront en caractère gras :

```
<body>  
<div id="rootline">rootline</div>  
<div id="header">  
    <div id="logo">logo</div>  
    <div id="headerimagetext">texteimageentete</div>  
</div>
```

Les termes nouveaux et les mots importants sont présentés en italique. Les mots que vous voyez dans les captures d'écran — dans les menus ou dans les boîtes de dialogue, par exemple — apparaissent ainsi dans le texte : « en cliquant sur la touche **Suivant**, vous arrivez à l'écran suivant. »

Toutes les entrées et sorties de lignes de commande sont affichées comme suit :

```
user@domain:/srv/www> ls -al htdocs/
total 38
drwxr-xr-x  6 user  group  512 May 23 02:42 .
drwxrwxr-x 14 user  group  512 Jul 24 17:51 ..
-rw-r--r--  1 user  group 4987 May 23 02:42 INSTALL.txt
-rw-r--r--  1 user  group  608 May 23 02:42 Package.txt
-rw-r--r--  1 user  group 8119 May 23 02:42 README.txt
...
```

Les commentaires des lecteurs

Les commentaires de la part de nos lecteurs sont toujours bienvenus. Faites-nous savoir ce que vous pensez de ce livre, ce que vous avez aimé ou ce que vous n'avez peut-être pas aimé. Les critiques de nos lecteurs sont importantes pour nous, dans la mesure où nous pouvons alors développer les sujets les plus pertinents pour vous.

Pour nous envoyer un commentaire général, vous pouvez simplement poster un email à l'adresse typo3@eyrolles.com, en vous assurant que vous mentionnez bien le titre du livre et l'objet de votre message.

Télécharger le code exemple pour le livre

Visitez le site <http://www.editions-eyrolles.com/>, et tapez le code **11781** dans le champ Recherche pour accéder à la page du livre. Vous y trouverez des liens vers les fichiers du code source des exemples du livre et vers d'autres ressources.

Les fichiers téléchargeables contiennent des instructions quant à leur utilisation.

Errata

Bien que nous ayons pris le plus grand soin pour assurer l'exactitude du contenu du livre, des erreurs peuvent toujours se produire. Si vous découvrez une erreur dans l'un de nos livres — cela peut être une faute dans le texte ou dans le code — nous vous serions reconnaissants de nous le signaler. En agissant de la sorte, vous pouvez épargner une certaine frustration aux autres lecteurs, et vous nous aidez également à améliorer les versions suivantes de ce livre.

Si vous trouvez des erreurs, signalez-les par mail à l'adresse typo3@eyrolles.com. Après les avoir vérifiés, nous ajouterons ces errata à la liste existante. Cette liste peut être consultée en consultant la page du livre sur le site <http://www.editions-eyrolles.com>.

Introduction

1

Chapitre

Introduction

Ce livre est basé sur la version originale allemande, dont différentes parties ont été mises à jour à plusieurs reprises, à cause de changements effectués sur le noyau de TYPO3 dans ses deux dernières versions (3.7.0 et 3.8.0).

Les auteurs aimeraient remercier tous leurs lecteurs pour leurs nombreux commentaires avisés, leurs conseils, et, bien sûr, pour le succès considérable qu'a rencontré l'ouvrage.

Nous voudrions aussi remercier la communauté qui nous soutient, à commencer par Kasper Skårhøj, les membres de l'association TYPO3, ensuite toutes les personnes appartenant à des groupes d'utilisateurs, à des listes mail, et enfin aux prestataires de service qui nous aident.

1.1 Historique

Kasper Skårhøj, né en 1976, travaillait déjà depuis la fin 1997 sur l'un des tout premiers systèmes de gestion de contenu pour la jeune entreprise « Superfish » à Copenhague. Après deux années de développement, un voyage à la conférence de Seybold à San Francisco, un certain nombre de projets à vocation pratique, et plusieurs rencontres (avec le « gourou »

d'Internet David Siegel), Kasper se rendit compte que « Superfish » n'était pas vraiment l'environnement adéquat ou idéal pour continuer à développer TYPO3.

Ceci s'explique en partie par le fait que « Superfish » s'orientait dans une nouvelle direction quant au type de service fourni. De plus, il y avait une raison qui n'est que trop familière à de nombreux développeurs de logiciels : la pression des délais pour sortir de nouvelles versions en vue du prochain salon, et la tendance générale à donner plus d'importance aux aspects visibles et à négliger les aspects invisibles, qui, à long terme, sont plus importants quand il s'agit de la qualité.

Les conséquences que Kasper tira de cette situation ne sont pas parmi les plus habituelles : alors que la plupart des développeurs d'Open Source de la première génération grandissaient dans un environnement universitaire — Linus Torvalds en étant un exemple de premier plan —, Kasper décida de renoncer à la sécurité d'une entreprise florissante, à laquelle il était de plus associé, pour travailler à plein temps sur sa conception du système de gestion de contenu. Un an plus tard, s'étant consacré uniquement à la mise en œuvre de cette conception, la version 1.5 fit son apparition. C'était en juillet 2000.

Le développement de TYPO3 resta un « one-man show » jusqu'en juillet 2002, avec cet avantage crucial que la qualité et la cohérence de l'ensemble demeurèrent à un très haut niveau.

Il y avait un désavantage : la progression du développement et un certain nombre d'éléments devaient, en quelque sorte, passer « à travers le chas de l'aiguille Kasper ». Après de nombreuses discussions avec les membres de la liste mail, suivies par une phase de travail frénétique, Kasper publia une nouvelle version en 2002, la version 3.5b1, qui, avec son gestionnaire d'extension, transforma immédiatement TYPO3 en un système modulaire. Depuis, la communauté a continuellement publié de nouvelles extensions, permettant d'élargir rapidement le champ d'application de TYPO3.

Référence 314624

Par conséquent, des groupes de travail ont été formés pour aborder tous les aspects de TYPO3. Kasper Skårhøj est au cœur du projet et est toujours responsable de la mise en circulation. Il travaille personnellement à la résolution d'un certain nombre de problèmes.

Il ne faudrait pas oublier que son engagement dépend essentiellement de la question suivante : pour lui, est-ce économiquement viable de se concentrer sur ces sujets ? C'est pourquoi les contributions à de grands projets commerciaux et les prestations de services effectuées à ces occasions, sont cruciales pour le projet TYPO3 : afin qu'il continue à pouvoir payer ses spécialistes les plus importants.

1.2 Qu'est-ce qu'un CMS ?

Les systèmes de gestion de contenu sont devenus incontournables pour créer et mettre en ligne du contenu sur Internet et les intranets. On appelle généralement les systèmes de logiciel de ce domaine « Web Content Management Systems » (WCMS), ou « Content Management Systems » (CMS). Lorsqu'ils sont intégrés dans une approche globale de l'information depuis la gestion des documents, jusqu'à la publication sur Internet, en passant par l'impression, on parle de gestion professionnelle de contenu.

1.2.1 Séparation du contenu de la forme

Un principe de base est la séparation du contenu de la forme. En pratique, cela signifie que la définition de la mise en forme est sauvegardée indépendamment du contenu en tant que tel, que ce dernier soit sous forme de texte, d'images ou d'autres formats. Ce principe comporte de nombreux avantages quand il s'agit de changer la mise en forme, ou de la protéger contre des auteurs de contenu prolifiques. Alors que le contenu peut être édité par les auteurs sans aucune influence sur les détails de l'affichage, la définition de la mise en forme peut être modifiée de façon indépendante, permettant donc des changements effectifs d'architecture, même au travers de très grands sites Web.

1.2.2 Cycle de vie du contenu

Les systèmes de gestion de contenu accompagnent le contenu dans toutes les étapes de son cycle de vie, depuis sa création avec les éditeurs, en passant par l'organisation des ressources, jusqu'au déploiement en ligne lors de la publication, et enfin à l'archivage des éléments de contenu.



Figure 1.1:
Cycle de vie du
contenu

Durant la phase de création, l'information est rassemblée et organisée globalement. Après la planification, l'information pertinente est mise par écrit, structurée quand c'est possible, et divisée en composants. Cette phase comprend également l'édition ainsi que des améliorations didactiques ajoutées par les auteurs.

L'étape d'organisation, ou de préparation, organise et sauvegarde le contenu selon un schéma qui convient à la fois aux résultats et aux sources d'information. De plus, on définit et on attribue ici les droits d'utilisateurs.

La publication a trait à la distribution et à la présentation du résultat sous la forme d'un site Web ou d'un produit, imprimé ou sous un autre format.

Les archives sauvegardent le contenu de manière à se prêter au mieux aux recherches, aux statistiques et, au bout du compte, à la réutilisation.

1.2.3 Système modulaire

TYPO3 est un système modulaire organisé en plusieurs couches. Depuis la sauvegarde jusqu'au déploiement, ces couches divisent le système en modules. Concernant la sauvegarde, vous disposez d'une API pour connecter TYPO3 aux différentes bases de données. Vous pouvez aussi enregistrer l'information dans des fichiers XML en utilisant les fonctions de la couche d'abstraction de la base de données. Par ailleurs, TYPO3 a défini une interface pour étendre le noyau du logiciel, nommé pour cette raison le système « d'extensions ». Ces extensions se connectent à TYPO3 via le gestionnaire d'extensions, qui sert aussi d'environnement de développement (IDE).

Les extensions peuvent ajouter des fonctionnalités à tous les niveaux et toutes les interfaces du logiciel, sans en changer le noyau, et assurant donc des mises à jour aisées et la stabilité du code source. Tout comme les vendeurs sérieux de gestion de contenu de logiciels, les systèmes Open Source et TYPO3 en particulier, ont adopté cette approche modulaire, même s'ils recourent à des concepts techniques différents, selon l'API disponible.

1.2.4 Groupe cible

Sur une période relativement courte, TYPO3 a pris, très rapidement, de l'ampleur. En termes de quantité, le groupe d'utilisateurs le plus représentatif est constitué d'entreprises de petite ou moyenne taille. En effet, la courbe d'apprentissage et la puissance de TYPO3 constituent un frein à son utilisation dans la sphère privée.

Référence 027059

Les entreprises capables de s'acquitter de leur droit d'entrée dans le monde de TYPO3, par le biais des services de spécialistes dûment qualifiés, peuvent, cependant, apprécier les avantages d'un logiciel libre en général, et de TYPO3 en particulier. Les raisons les plus souvent invoquées comprennent la licence (GPL), la qualité, la documentation considérable, la communauté qui s'accroît rapidement, et les nombreuses recommandations de la part d'entreprises et d'organisations renommées. Une sélection de ces dernières est publiée sur TYPO3.com, à la référence ci-contre.

Parmi les quelque 200 000 installations reprises dans le répertoire d'extensions de TYPO3 entre juillet 2002 et le début de 2005, on trouve un grand nombre de noms bien connus et de marques célèbres qui ont fait confiance à TYPO3, prouvant la validité des systèmes Open Source d'entreprise en les utilisant dans leurs intranets et dans d'autres applications critiques.

1.2.5 Références TYPO3

Référence 589606

La liste des références dans TYPO3.org présente une sélection de projets types basés sur TYPO3. Parmi les prestataires de services, il existe un grand nombre de noms réputés venant d'Allemagne et du reste de l'Europe, mais également beaucoup d'entreprises de petite ou moyenne taille, qui voient là une opportunité d'entrer en compétition à un niveau technique avec de grands fournisseurs, proposant des prix attractifs et des délais courts pour la réalisation de projets.

La licence GPL : « GNU General Public License »

La licence publique générale GNU est la licence Open Source la plus largement utilisée. Cependant, son contenu, apparemment simple, continue à provoquer des contradictions dans différents pays et dans différentes législations. Une opinion légale en suit une autre et, parce que le logiciel est distribué gratuitement, des controverses apparaissent en ce qui concerne la sécurité du consommateur, les règlements à propos des autorisations, et les intérêts des programmeurs, qui peuvent s'avérer significatifs, selon le champ d'application et dans la mesure où l'utilisation du logiciel se révèle critique pour l'entreprise. Il est donc conseillé de se renseigner à propos du contenu de la GPL et ses conséquences légales, dans le pays où le logiciel est utilisé. Quand cela se révèle approprié, les garanties offertes par une agence ou un prestataire de services techniques, qui prennent la relève ou travaillent à la mise en œuvre, doivent être réglementées.

Les extensions revêtent un rôle particulier. Aussi longtemps qu'elles ne fonctionnent pas comme des applications indépendantes, elles sont sujettes, elles aussi, à la GPL.

Une fonction profondément ancrée dans le code source de TYPO3, qui est normalement développée sous la forme d'une extension, et qui n'est pas juste une adaptation ou un script basé sur un programme existant, est Open Source, et est elle aussi soumise aux conditions GPL.

La licence GPL spécifie que l'auteur n'est pas obligé de publier, mais ne peut pas non plus empêcher la publication une fois qu'elle aura circulé et sera peut-être arrivée jusqu'à un client. L'auteur ne peut pas non plus opposer des restrictions aux modifications, ni à de futures modifications effectuées par des tiers, une fois qu'ils sont entrés en possession du code.

En ce qui concerne les intérêts économiques, des problèmes peuvent surgir concernant l'utilisation de TYPO3, pour certains éditeurs habitués à penser en termes de produits commercialement viables.

Mais, par analogie, il faut garder à l'esprit que TYPO3 a été développé sans licence et sans coût.

De fait, la communauté dans notre cas réagit de deux façons différentes.

De nombreuses extensions ne sont pas publiées ; sur les clés d'extension nécessaires à l'échange via TYPO3.org, un cinquième seulement sont des extensions publiées. Ceci n'inclut pas les nombreuses extensions qui ne sont pas enregistrées, mais inclut en revanche beaucoup d'autres inachevées, et d'autres trop spécialisées pour justifier leur publication.

Il est certainement très peu probable que des clients publient eux-mêmes les extensions qu'ils auraient obtenues d'un prestataire.

Beaucoup de spécialistes se servent de plusieurs projets pour créer et contribuer à développer une extension, jusqu'à ce que leur investissement en temps et en connaissances soit rémunéré, et alors seulement ils publient leur travail, quand celui-ci est susceptible d'accroître leur renommée.

1.3 La communauté TYPO3

La communauté TYPO3, composée d'utilisateurs et d'enthousiastes, grandit rapidement et devient de plus en plus internationale.

Le noyau de la communauté est représenté par des listes mail — la liste mail anglaise principale est la liste de référence, à laquelle on peut aussi s'abonner via des newsgroups. Référez-vous à **TYPO3.org** pour plus de détails à propos des listes mail.

La communauté assure un support, comme dans la plupart des projets Open Source, en offrant une aide rapide et pratique, même pour les problèmes plus compliqués techniquement parlant, tant que les règles habituelles du choix de nom des sujets ainsi qu'une formulation précise des questions sont respectées. Il est important de lire les archives, les FAQ, et les documents d'aide aux débutants, de façon à ne pas encombrer les listes avec des questions ayant déjà reçu de nombreuses fois une réponse, ce qui peut ennuyer les utilisateurs expérimentés. Souvenez-vous que l'aide que vous recevez est volontaire et gratuite, et que vous n'y avez pas automatiquement droit.

Référence 139514 Si quelqu'un a besoin d'une plus grande disponibilité, de matériel pour s'exercer, ou bien a des questions plus complexes à poser, il peut entrer en contact avec l'un des nombreux prestataires de services qu'on trouve dans la section « Consultancies » à l'adresse **TYPO3.com**. Ces entreprises ont été sélectionnées d'après la compétence dont elles ont fait preuve dans leurs projets.

Référence 424461 Quelqu'un qui a déjà construit un savoir-faire pertinent en matière de TYPO3 souhaitera peut-être se rendre utile en développant davantage la documentation existante, ou en contribuant au projet d'une autre façon. Le projet vit grâce à de telles contributions à tous les niveaux, de sorte que l'aide et le retour d'information sont réellement bienvenus, et font l'objet de longues discussions. Si vous désirez faire cela, il est important d'avoir une impression générale au préalable : quels sujets ont déjà été discutés, et avec quels résultats. Dans ce but, nous vous conseillons de visiter les archives des listes mail sur **TYPO3.org**, dans le champ **Documentation** → **Mailing-Lists**. Selon le sujet, vous devrez peut-être regarder dans les listes « Developer », « Marketing », ou une autre liste mail.

Ensuite, vous devrez vous assurer qu'il n'existe pas de projet à l'objectif similaire, ou qui recoupe le vôtre.

Dans **TYPO3.org**, dans la section **Développement** → **Projets**, vous trouverez une liste de tous les projets en cours et des secteurs de travail, avec les partenaires à contacter.

L'une des caractéristiques de TYPO3 et de la communauté est que beaucoup des participants ont déjà fait connaissance, lors de la rencontre annuelle pour la plupart d'entre eux, le « TYPO3 ! Snowboard Tour », qui attire de plus en plus de participants chaque année.

En dehors des discussions, des soirées de travail intensif et des nombreuses questions qui trouvent une réponse, ainsi que des projets qui y sont lancés, l'objectif est aussi d'apprendre à connaître les personnes qui se trouvent derrière les adresses mail, et bien entendu également d'apprécier les sports d'hiver ensemble. Donc, quiconque souhaite combiner l'apprentissage d'un savoir-faire intensif avec des vacances trouvera difficilement une meilleure opportunité que cet événement.

De par l'utilisation considérable de TYPO3 dans des organisations et la très large communauté d'utilisateurs, la première conférence internationale devait bien avoir lieu un jour : ce fut TyCON3, en septembre 2005, à Karlsruhe, en Allemagne. Pour plus d'informations, vous pouvez visiter le site Web suivant :

<http://tycon3.typo3.org>

1.4 L'association TYPO3

En novembre 2004, un groupe issu de la communauté TYPO3, y compris Kasper Skårhøj et d'autres personnes ayant contribué au projet depuis longtemps, ont préparé et fondé une association appelée l'Association TYPO3. Son but principal est de soutenir le développement du noyau sur une base plus stable, et d'améliorer la transparence et l'efficacité de divers aspects du projet TYPO3.

1.4.1 Objectifs

- L'organisation d'événements au bénéfice de l'information et de la formation de ses membres
- La communication avec les membres et avec le grand public, en vue de promouvoir et d'accroître la connaissance et la compétence relatives à l'utilisation du logiciel TYPO3, particulièrement en vertu de son projet de site Web
- La formation et la certification, assurant la qualité du service
- Favoriser le développement de TYPO3
- Supporter l'adaptation des logiciels internationaux standards au sein de TYPO3
- La représentation des membres
- Les relations publiques et les activités qui contribuent à développer la connaissance et l'utilisation du logiciel TYPO3.

1.4.2 Être membre et s'inscrire

L'Association TYPO3 compte deux types de membres :

Membres actifs

Les membres actifs sont des personnes qui ont travaillé en accord avec TYPO3 et sont à la fois désireux et capables d'assister aux assemblées générales, où ils ont un droit exclusif de vote et de décision quant au futur de l'Association. Les membres actifs sont nommés en fonction de leur mérite, et doivent être recommandés par deux membres actifs, ou par un cinquième de l'assemblée.

Membres de soutien

Être membre de soutien est possible pour tout le monde, mais requiert une inscription formelle. L'Association propose une fiche d'inscription en ligne et un moyen de paiement des droits d'entrée en tant que membre à l'adresse <http://association.typo3.org>. Les membres sont soit des personnes, soit des entreprises, et il leur est demandé de soutenir les buts et objectifs de l'Association, à travers leurs actions.

Référence 394945

Tous les membres auront le droit de se réclamer de leur qualité de membre exclusivement, et donc de soutenir TYPO3, uniquement si leur admission en tant que membres a été acceptée.

Organismes et travail pratique

L'Association se compose des organismes suivants :

Conseil (board)

Le conseil est l'organisme exécutif qui assure la gestion quotidienne, les questions légales et la comptabilité.

Assemblée générale

L'AG est la plus haute autorité et élit le conseil, contrôle le travail de ce dernier et, d'une façon générale, décide de toutes les questions d'importance lors de son meeting annuel. L'AG est ouverte à tous les membres, mais seuls les membres actifs ont le droit de vote.

Comités

Les comités sont les groupes qui effectuent le travail en tant que tel quand il s'agit de discuter de l'utilisation des fonds, de la communication au sein de la communauté, des formations, des événements, et ainsi de suite. Légalement, ils sont nommés par le conseil et l'AG. Ils font des recommandations, qui sont ensuite exécutées par le conseil.

2

Chapitre

Installation

Dans ce chapitre, nous abordons les différents types d'installation de TYPO3 et leur champ d'application. Nous décrivons ensuite les étapes nécessaires à leur mise en œuvre. Si on se fie au nombre d'installations déjà existantes, les obstacles à l'installation de TYPO3 ne semblent pas infranchissables. Toutefois, les auteurs de cet ouvrage pourraient parler des heures durant de leurs premiers pas en évoquant certains problèmes susceptibles de dérouter le débutant.

Pour terminer, vous trouverez un aperçu de quelques ressources mises à votre disposition pour résoudre les problèmes que vous risquez de rencontrer.

2.1 Choix du packaging TYPO3 et de la configuration du serveur

TYP03 requiert simplement une base de données et un serveur Web configuré avec PHP ; sur cette base, le matériel utilisé, le système d'exploitation, le système de base de données ainsi que le serveur Web sont choisis en fonction d'un grand nombre de critères, dont nous reprenons ici les plus importants.

Le choix du système d'exploitation est déterminant. TYPO3 fonctionne sur la plupart des systèmes de type UNIX, ainsi que sur Windows. Il n'y a pas de différence entre ces deux versions en termes de fonctionnalités de base, même si certaines extensions exigent des programmes UNIX. Consultez la documentation sur les extensions concernées afin de tenir compte de leurs restrictions.¹

Un des avantages techniques de l'utilisation d'un système UNIX est la rapidité des mises à jour grâce aux « liens symboliques ».²

Savoir quel système d'exploitation est majoritairement utilisé au sein de la communauté d'utilisateurs de TYPO3 est aussi un facteur important. La tendance, du moins en termes de quantité, est à l'usage des systèmes Linux. Dès lors, le support et, pour certaines extensions, les nouveaux développements qui dépendent d'un système d'exploitation sont plus fréquents pour Linux. S'il n'y a aucune contre-indication — telle qu'une infrastructure déjà existante basée sur Windows ou votre (in-)expérience —, Linux est le meilleur choix pour l'exploitation de TYPO3.

Les chapitres suivants abordent de manière plus approfondie le choix de logiciels et de matériel informatique en ce qui concerne le serveur, le système de base de données, d'autres logiciels utiles, et finalement, le choix du packaging de TYPO3.

2.1.1 Matériel

Basé sur PHP, TYPO3 requiert au minimum un matériel informatique supportant un serveur Web. Même si un vieux 286 avec 32MB de mémoire vive conviendrait à cet effet, il ne constitue pas une plate-forme suffisamment puissante pour exploiter TYPO3. Le système doit avoir une mémoire vive de 512 MB ou plus afin de fournir les performances requises.

Si vous désirez utiliser votre propre serveur, considérez les facteurs suivants pour dimensionner votre matériel :

Type d'utilisation

Le facteur décisif est la manière dont l'information disponible sur votre site Web sera utilisée : le site sera-t-il statique, en tout ou en partie ? Désirez-vous gérer un site portail ? Voulez-vous créer du contenu dynamique sur votre serveur Web ? Prévoyez-vous de mettre en ligne une application telle qu'une boutique, des cartes virtuelles ou des forums ? La règle suivante prévaut : au plus nombreuses sont les fonctions effectuées par le serveur, au plus élevés sont les besoins en matériel et les dépenses pour séparer le système en ligne du système de production.

Charge du serveur

Un certain nombre de paramètres peuvent être quantifiés :

- Combien d'utilisateurs utiliseront le système simultanément ?
- À quelle vitesse (en secondes) une page doit-elle être servie par le serveur ?

¹**Indexed Search**, par exemple, est un puissant moteur de recherche utilisé dans TYPO3 nécessitant un logiciel UNIX pour indexer les documents. Il en est de même pour plusieurs fonctionnalités de l'extension DAM et pour les extensions qui manipulent le format PDF.

²Sous Windows, l'installation du progiciel complémentaire *Junction* permet d'utiliser les liens symboliques. Voir Référence 394 945.

- Combien de pages par heure et par mois doivent être délivrées ?
- Quel est le volume mensuel de trafic prévu, en MB ou GB ?

Il sera difficile de prendre une décision fondée si ces grandeurs ne sont pas connues. Si vous avez des doutes, essayez de vous informer auprès des prestataires offrant des services semblables aux vôtres afin d'avoir une meilleure estimation. Il existe trois scénarios typiques :

1. Charge faible à moyenne

Il s'agit ici de sites n'ayant aucune raison économique justifiant le recours à un serveur dédié. Pour ces sites, on ne prévoit pas plus de 10 visiteurs à la fois dans le backend et 50 dans le frontend, une génération de page en 1,5 secondes maximum, pas plus de 100 pages délivrées par heure et moins de 100 000 pages servies par mois. En tout, le trafic mensuel ne dépasse pas 5 GB par mois. Si votre profil correspond à celui-ci sur plusieurs points, vous avez tout intérêt à sélectionner un hébergeur de qualité, qui idéalement propose des solutions où TYPO3 est déjà pré-installé.

Si vous optez pour une solution où TYPO3 n'est pas encore installé, assurez-vous d'abord que les conditions d'installation suivantes sont remplies :

- au moins 100 MB d'espace disque
- une base de données MySQL
- PHP à partir de la version 4.3.x avec les bibliothèques GDLib et Freetype
- ImageMagick
- accès via SSH

À ce stade, un avertissement s'impose : depuis longtemps, la location d'une infrastructure pour le Web ne requiert plus des montants astronomiques. Via des offres à bas prix pour des serveurs dédiés, n'importe qui disposant de quelques euros peut mettre en place en moins d'un mois une offre de services en tant qu'hébergeur ou de fournisseur d'accès à Internet. La connaissance et la compétence en la matière ne sont pas indispensables, ce qui aboutit parfois à de fâcheuses conséquences.

Le secteur d'accès à Internet est soumis aux mêmes contraintes économiques que tout autre secteur d'activité, et de trop bas prix résultent d'un des points suivants :

1. Une mauvaise connaissance du matériel : quelqu'un d'averti sait qu'une courte période d'indisponibilité d'un serveur peut détruire le fruit d'un long travail ;
2. Peu de frais en personnel : votre prestataire sous-traite son infrastructure chez un des grands fournisseurs du marché (ce qui implique peu de marge de manœuvre pour votre prestataire) ;
3. Taille du prestataire (un grand prestataire pourra plus facilement engager des experts).

Bien sûr, il est parfaitement justifié de rechercher l'offre la plus avantageuse.

Seulement, avec des prix écrasés, vous ne pouvez pas vous attendre à un service optimal et à une protection des données respectant l'installation et le contenu de votre CMS.

Par prudence, pour garantir un niveau de sécurité minimal (ainsi qu'un fournisseur qui est en bonne santé financière, redevable contractuellement d'indemnités en cas de problème), vous devriez payer un prix acceptable, c'est-à-dire un minimum de 25 euros par mois. À ces

conditions, vous pouvez faire appel à des hébergeurs de plus petite taille qui vous offrent un service fiable et personnalisé.

Indépendamment du budget de votre projet, exigez toujours une sauvegarde automatique, faites-en un préalable lors de la sélection d'un prestataire.

2. Charge moyenne à élevée et Intranet

Il est judicieux d'avoir votre propre serveur si votre application CMS en justifie l'installation et l'exploitation. Dans ce cas, le coût de la location du serveur ou de la co-location devient marginal par rapport aux autres coûts du projet.

Plusieurs études ont démontré qu'une proportion de 35% des coûts était consacrée à l'achat ou la location d'un serveur contre 65% pour la maintenance et les autres coûts associés.

Pour l'intranet, vous n'avez pas d'autre choix que d'avoir votre propre serveur. La question centrale est ici celle du dimensionnement. Avant toute décision, il est fortement conseillé de procéder à une évaluation des besoins, en tenant aussi compte de leur évolution (la maintenance de deux serveurs ne représente pas plus de 10% de travail supplémentaire que pour un seul serveur).

La formule la plus répandue pour le calcul des coûts sur la durée de vie d'un investissement, appelé coût total de propriété ou encore *Total Cost of Ownership*, en anglais (TCO), peut être par exemple présentée comme suit :

L' « iceberg » du TCO

Coûts apparents

- + investissement
- + achat de logiciels et de matériel
- + installation
- + mise en œuvre

Coûts cachés

- + adoption du système
- + support
- + administration
- + formation
- + maintenance
- + degré de non-utilisation

= coût total de propriété

Ce mode de calcul s'est considérablement affiné au cours du temps³, mais pour un exemple ou pour une première évaluation des coûts d'une opération, la liste reprise ci-dessus suffit.

³cf. Bensberg/Dewanto : « TCO VOFI for eLearning Platforms », <http://www.campussource.de/org/opensource/docs/bensbergVor.doc.pdf>

Si, en fonction de votre calcul, vous optez pour un serveur dédié, il est essentiel de clarifier les points techniques suivants : les performances du matériel, les possibilités de le mettre à jour, la disponibilité de pièces de rechange, la stratégie de sauvegarde ainsi que la sécurité (droits d'accès, etc.).

3. Charge élevée

Pour une charge élevée, c.-à-d. plus d'un million de pages servies par mois, la solution standard est de combiner plusieurs serveurs en *cluster*. La motivation de ce choix est de répartir la charge ainsi que d'assurer une redondance du système. Suivant le rôle qu'on lui assigne, la charge maximale à laquelle fait face le serveur peut être dépassée même avec un trafic peu élevé. De plus, certaines conditions d'utilisation de l'application —par exemple dans un système informatique de gestion— ne tolèrent aucune interruption. Les serveurs Web et les serveurs de base de données peuvent aussi être administrés et dimensionnés séparément, en fonction de la charge qu'ils ont à supporter. Divers scénarios existent, allant de systèmes de 10 serveurs Web Apache, installés chacun avec leur propre matériel et un serveur MySQL, jusqu'à des systèmes composés de 3 serveurs Apache et de 5 serveurs MySQL.

Dans tous les cas, il est ici question de solutions spécifiques qui nécessitent une planification et une implémentation précises, ainsi qu'un suivi constant.

2.1.2 Serveur Web

En théorie, TYPO3 pourrait être exploité avec Apache, IIS ou tout autre serveur utilisant PHP. La combinaison la plus utilisée, en termes de quantité, est la combinaison Apache-PHP. La version de PHP et sa configuration ont une influence sur TYPO3 ; ce phénomène est décrit à la rubrique « Server Compatibility » sur le site TYPO3.org (cf. référence).

Référence 504537

2.1.3 Base de données

La base de données standard pour TYPO3 est MySQL. Elle est restée longtemps la seule base de données supportée. Depuis lors, une couche d'abstraction de base de données (DBAL⁴) a été introduite. Comme cette couche utilise un langage SQL compatible avec MySQL au lieu d'un métalangage, MySQL reste la solution la plus performante, puisque dans ce cas, les requêtes vers la base de données ne doivent pas être transformées. De plus, plusieurs extensions utilisent leurs propres requêtes. Des vérifications sont donc nécessaires au cas par cas si l'on veut faire fonctionner une extension avec d'autres bases de données pour s'assurer que le code gérant les requêtes reste compatible.

Choisir un système SGBDR (Système de Gestion de Bases de Données Relationnelles) autre que MySQL est une décision qui doit être prise avec précaution ; au bout du compte, les coûts d'installations et de mises à jour deviendront, à long terme, supérieurs à ceux d'une installation standard.

Une variante intéressante de l'utilisation d'autres bases de données est de combiner différentes bases de données, afin de conserver les données spécifiques sur votre propre système de base de données et de les rendre disponibles pour une application spécifique utilisée dans TYPO3.

⁴DBAL est l'abréviation de *Database Abstraction Layer*.

L'intégration directe de données à partir d'autres systèmes de bases de données est alors rendue possible, évitant du même coup les désavantages de la réplication et de la synchronisation de données. L'abstraction de base de données n'est pas réservée qu'aux systèmes SGBDR : les fichiers plats tels que les données XML peuvent aussi être interrogés via des requêtes SQL.

Référence 613803

Nous n'envisageons ici que la situation standard avec MySQL, l'information sur d'autres scénarios n'étant pas encore disponible. Toutefois, la documentation sur l'*abstraction de base de données* (DBAL) fournit quelques indications (cf. référence).

2.1.4 Autres logiciels

Deux bibliothèques sont nécessaires à TYPO3 pour la manipulation d'images. Ces logiciels utilitaires sont entièrement optionnels ; TYPO3 fonctionne aussi sans traitement d'images et sans ces bibliothèques. La première bibliothèque est *GDLibrary*, une extension PHP, qui peut être complétée par *Freetype*, une extension offrant des fonctions pour représenter des polices de caractère. Puisque GDLibrary est déjà compris dans une installation PHP standard, nous ne nous y attarderons pas ici.

Référence 353034

La seconde bibliothèque, utilisée principalement pour le redimensionnement et la création des fichiers image en pré-visualisation, est *ImageMagick*. L'utilisation d'une ancienne version (version 4.2.9) est recommandée pour TYPO3, cette dernière présentant des avantages par rapport aux versions plus récentes.⁵

TYPO3 peut être utilisé avec les dernières versions de ImageMagick si les inconvénients pour les fonctions de masquage, de contour et de lissage n'affectent pas votre site Web.

Référence 436028

La version recommandée (4.2.9) est disponible sur le site de TYPO3.org sous la référence indiquée ci-contre. Vous pouvez vous procurer la dernière version d'ImageMagick soit par le gestionnaire de paquetage de votre distribution Linux, soit en le téléchargeant à partir du site d'ImageMagick.⁶

Vous pouvez aussi utiliser *GraphicsMagic*, une bibliothèque basée sur le projet ImageMagick, qui entend maintenir le développement de l'API dans une ligne plus stricte.⁷

2.1.5 Choix du paquetage TYPO3

TYPO3 existe en plusieurs paquetages, selon l'usage auquel on le destine. Leur format dépend du système d'exploitation :

Tous les paquetages UNIX sont des archives de type tar⁸ et se terminent par l'extension *.tar.gz*. Les paquetages Windows sont des archives de type zip, et se terminent par l'extension *.zip*.

Il n'y a aucune différence réelle entre les fichiers contenus dans les deux types d'archive. La seule différence est la taille du paquetage. Le paquetage tar.gz occupe moins d'espace disque

⁵Nous n'avons pas reçu de réponse satisfaisante lorsque nous avons interrogé l'équipe de développement d'ImageMagick, concernant la baisse de qualité dans certains domaines. L'échange d'emails reprenant les désavantages de chaque version d'ImageMagick est disponible à la référence citée plus haut.

⁶<http://www.imagemagick.com/>

⁷<http://www.graphicsmagick.org/> : l'évolution inconstante du développement de l'API d'ImageMagick a poussé plusieurs développeurs à se passer de cette bibliothèque. TYPO3 s'est adapté à ce comportement capricieux en ajoutant des options de configuration pour tenir compte des particularités de chaque version d'ImageMagick.

⁸Une archive tar est le nom donné aux archives créées à l'aide du programme tar (qui est en général aussi comprimé avec gzip). Ce format accepte les liens symboliques.

que le paquetage .zip car les liens symboliques évitent de dédoubler certains dossiers. Le paquetage .zip contient ces dossiers en double exemplaire, comme le montre la capture d'écran.

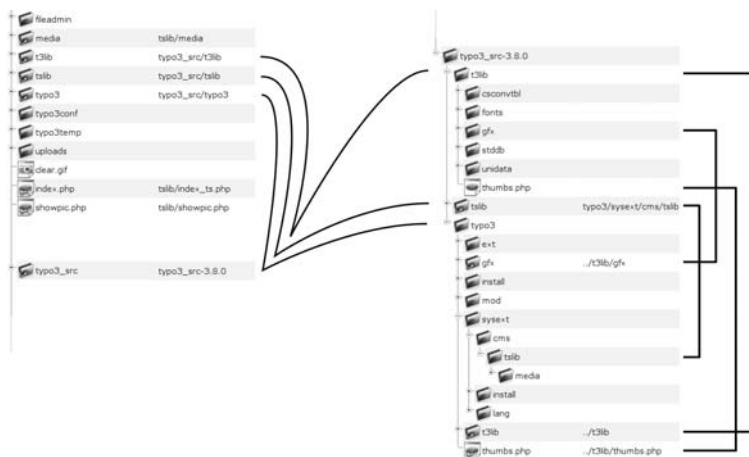


Figure 2.1:
à gauche : liens
symboliques et leur
relation vers les
fichiers du répertoire
src ; à droite : liens
symboliques vers les
répertoires sources à
l'intérieur du
répertoire src

Des raisons historiques expliquent le double usage des répertoires (à l'aide de liens symboliques). Notez que les liens symboliques ne peuvent pas être créés via un accès FTP. Si votre accès au serveur Web est limité à FTP, sélectionnez un paquetage zip.

Depuis peu, à l'aide de logiciels utilitaires tels que *Junction*, il est aussi possible de créer des liens dans le système de fichiers d'une partition NTFS de Windows.

Référence 394945

Les différents paquetages comprennent toujours la version la plus récente de TYPO3 ; ils ne diffèrent que par les exemples qu'ils contiennent. En général, les paquetages tar.gz ne contiennent pas le code source de TYPO3, repris dans le fichier typo3_src suivi du numéro de la version, qui doit être téléchargé séparément. Par contre, tous les paquetages zip contiennent le code source.

Les principaux paquetages TYPO3 sont les suivants :

QuickStart

Cette version est prévue pour les néophytes et comprend le didacticiel du débutant. Vous devriez sélectionner ce paquetage si vous voulez suivre le didacticiel car il contient tout le matériel et les données pour les exemples. Il est aussi disponible sous la référence ci-contre.

Référence 995697

Test Site

Le paquetage Test Site contient des exemples de gabarits TypoScript, différents types de menus et des extensions telles qu'une boutique, un annuaire et des actualités. Ce paquetage, bien qu'il soit un peu dépassé, est utile si vous souhaitez tester ou découvrir le système par vous-même à l'aide d'exemples.

Dummy

Le paquetage Dummy diffère des paquetages Quickstart et Test Site par sa base de données vide et par l'absence d'exemples. Il sert de paquetage standard pour les développeurs qui commencent un nouveau projet en partant de rien.

La copie de la base de données contient un compte d'administrateur, auquel on a accès avec le nom d'utilisateur `admin` et le login `password`.

TYPO3 Source

Le paquetage TYPO3 source est une archive de type tar qui contient tous les répertoires nécessaires au fonctionnement du système de base. Les paquetages zip contiennent par défaut ce paquetage.

Les liens symboliques permettent d'opérer plusieurs installations TYPO3 (sites Web) à partir d'une source unique. Les extensions peuvent être stockées dans le répertoire global d'extensions `typo3/ext/` ou dans le répertoire `typo3conf/ext/` propre à une instance de site Web.

2.2 Installation de test et d'initiation

Référence 056013

TYPO3.org offre toute une gamme de paquetages adaptés aux différents systèmes d'exploitation. Nous approfondissons ici les deux distributions classiques : le programme d'installation **WAMP** (Windows Apache MySQL PHP) conçu pour Windows par Ingmar Schlecht, et le paquetage Quickstart pour Linux. Il existe d'autres paquetages et guides d'installation pour BigApache, Mac OS X, Debian, Gentoo, Mandrake, etc. En jetant un œil sur la section « download » et sur la matrice de documentation du site TYPO3.org, vous trouverez de nouvelles idées pour l'utilisation de votre installation.

2.2.1 Installation WAMP sous Windows

Vous trouverez le programme d'installation WAMP parmi les paquetages sur la page de téléchargement de TYPO3.org. Il est très simple d'utilisation :

1. Téléchargez le programme sur votre ordinateur.
2. Ouvrez le fichier par un double-clic.
3. Après un court laps de temps, une boîte de dialogue s'affiche pour vous demander d'accepter la licence GPL. Elle constitue une licence relativement courte, que vous pouvez lire si vous désirez connaître vos droits. Si vous ne l'acceptez pas, vous êtes temporairement arrivé à la fin de votre carrière TYPO3 : il n'existe pas d'autre licence pour TYPO3, et vous devez absolument adhérer à ses conditions si vous ne voulez pas en perdre le droit d'utilisation.
4. Un message apparaîtra ensuite pour vous informer que le programme écrira tous les fichiers dans le répertoire `C:\apache`. Acceptez seulement si vous n'avez pas de répertoire préexistant portant ce nom, ou si vous souhaitez effacer toutes les données qu'il contient. En effet, toutes les données précédemment sauvegardées dans ce dossier seront irrémédiablement perdues.
5. L'installation est terminée.

Une nouvelle entrée **TYP03** se trouve maintenant dans votre menu **démarrer** avec les options suivantes :

Start Apache

Lance le serveur Web Apache

Start MySQL

Lance le serveur de base de données MySQL

Stop MySQL

Arrête le serveur de base de données MySQL

TYP03 start Servers before

Appelle le frontend de TYP03 dans une fenêtre de votre navigateur à l'adresse <http://localhost>

TYP03 (Alternative URL)

appelle TYP03 à l'adresse <http://127.0.0.1>

Si vous lancez Apache et MySQL et appelez la page de démarrage TYP03, vous trouverez toute l'information au sujet du frontend et du backend. Le programme d'installation WAMP s'est alors déjà chargé de toutes les autres opérations d'installation, et TYP03 est prêt pour les tests, ainsi que pour vos propres essais de programmation.

2.2.2 Installation Quick Install sous Linux

Si vous avez un serveur Web disponible qui supporte MySQL et PHP, vous installerez très facilement TYP03 sur Linux ou sur d'autres systèmes d'exploitation de la famille UNIX tels que BSD, OS X, etc.

Si vous avez un accès SSH au serveur, connectez-vous et téléchargez l'archive de Quickinstall à partir du site TYP03.org à l'aide de `curl` ou `wget`. Les adresses sont reprises sur la page de téléchargement de TYP03.org.

Référence 056012

Si vous n'avez qu'un accès FTP sur votre serveur Web, utilisez le paquetage zip. Transférez tous les fichiers sur le serveur dans le répertoire source de votre site Web (ils sont souvent appelés `.../htdocs/`, `.../html/` ou `.../www/`).

Vous donnez maintenant des droits d'accès en écriture sur certaines parties de TYP03. Si vous accédez au serveur uniquement en FTP, vous devrez procéder autrement. Plusieurs serveurs Web ont des interfaces pour l'édition de fichiers tels que **Cpanel**, **Confixx** ou **Webmin**. En cas de doute, contactez votre administrateur ou votre fournisseur.

Les droits d'accès des fichiers suivants doivent être modifiés comme indiqué ci-après :

```
chmod 777 typo3/temp/
chmod 777 typo3/ext/
chmod 777 typo3temp/
chmod 777 typo3conf/
chmod 777 typo3conf/ext/
chmod 777 uploads/
chmod 777 fileadmin/
```

Attention : en étendant ainsi les droits d'écriture à l'ensemble des utilisateurs, la sécurité de votre système peut être mise en danger par quiconque ayant accès à votre serveur. Vous pouvez

consulter la section 2.3.1 pour plus d'informations sur la manière de sécuriser votre environnement.

Appelez TYPO3 à l'adresse `http://www.votredomain.com/index.php` de votre serveur Web.

Si vous n'avez pas de fichier `index.html` dans le répertoire de votre serveur Web, vous ne devez pas entrer `index.php` à la fin de l'URL. Sinon, nous vous suggérons de renommer `index.html` en `index_alt.html`. Vous pourrez bien sûr l'effacer à partir du moment où vous n'en avez plus besoin.

Avant de basculer dans le mode « 1-2-3 » de l'outil d'installation, un message d'avertissement s'affiche, vous demandant de modifier immédiatement le mot de passe de l'outil d'installation.

Entrez un nom d'utilisateur et un mot de passe pour la base de données MySQL et soumettez le formulaire. Ensuite, vous pourrez créer une nouvelle base de données, ou en sélectionner une préexistante. Dans ce dernier cas, n'oubliez pas que toutes les données qu'elle contient seront écrasées !

Chargez le fichier `quickinstall.sql` dans la base de données. Vous pouvez maintenant vous connecter au backend avec le nom d'utilisateur `admin` et le mot de passe `password` à l'adresse `http://www.votredomaine.com/typo3`. Vous pouvez accéder à l'outil d'installation avec le login `joh316`.⁹ Veillez à changer le mot de passe de l'outil d'installation dès votre première connexion !

L'installation de base est maintenant terminée.

2.3 L'installation en production

L'installation en production se caractérise principalement par l'utilisation d'un packaging ne contenant que la source (sans exemples), et occupant de ce fait un espace disque minimum. De plus, l'outil d'installation contient un grand nombre d'options supplémentaires qui ne sont pas incluses dans le mode simplifié d'installation. C'est un point essentiel, non seulement pour optimiser la configuration du système, mais aussi pour avoir un aperçu de la structure de TYPO3.

2.3.1 Installation LAMP

En plus d'être le système d'exploitation le plus utilisé pour les serveurs Web, UNIX/Linux est idéal pour l'utilisation de TYPO3. La plate-forme Linux est très avantageuse, particulièrement en ce qui concerne les mises à jour ; par ailleurs, il existe plusieurs logiciels utilitaires, indispensables à certains modules additionnels de TYPO3, qui ne sont disponibles que sous Linux. C'est pourquoi il est recommandé de lire la documentation en ligne à propos des extensions avant de les installer. Par la suite, nous envisageons une installation standard avec MySQL.

⁹Pour les curieux : Kasper Skårhøj fait ici référence au verset de l'évangéliste Jean : « Car Dieu a tant aimé le monde qu'il a donné son Fils unique : ainsi tout homme qui croit en lui ne périra pas, mais il obtiendra la vie éternelle ».

En fonction des distributions Linux, des différences pourraient apparaître entre l'installation avec Apache/PHP et avec MySQL. ImageMagick est nécessaire au traitement et au dimensionnement des images. Vous pouvez aussi utiliser GraphicsMagick, une autre bibliothèque de manipulation d'images développée sur la base d'ImageMagick (voir page 36). Consultez la documentation de votre système ou le gestionnaire de paquetage approprié. Dans tous les cas, PHP doit avoir au moins 16 MB de mémoire (voir le fichier `php.ini`) et permettre le chargement de fichiers de grande taille (configuré dans `php.ini` et Apache).

Pour installer TYPO3, vous avez d'abord besoin d'une base de données MySQL vide avec son nom d'utilisateur et son mot de passe, ainsi qu'un des paquetages décrits plus haut, disponibles sur le site TYPO3.org.

Vous téléchargez ensuite la source et le paquetage dummy, qui contient des liens symboliques et un répertoire de configuration, ce qui vous évitera de saisir quelques lignes de commandes. Votre navigateur est la meilleure façon d'ouvrir la page de téléchargement du paquetage qui vous intéresse. Ouvrez en même temps une application de commande (p.ex. *Bash*, *Term*, *Putty*) et connectez-vous au serveur Web via SSH :

```
~$ ssh user@domain.com
```

Passez au répertoire de votre serveur Web, situé un niveau au-dessus du répertoire du site Web :

```
user@domain:~> cd /srv/www
```

Téléchargez la version actuelle des paquetages dummy et source. L'adresse correcte et le nom du dossier sont indiqués à la page de paquetages sur le site TYPO3.org (cf. référence) :

Référence 056011

```
user@domain:/srv/www> wget \
> http://typo3.sunsite.dk/unix-archives/3.8.0/dummy/dummy-3.8.0.tar.gz
```

et

```
user@domain:/srv/www> wget \
> http://typo3.sunsite.dk/unix-archives/3.8.0/dummy/dummy-3.8.0.tar.gz
```

Ici, `/srv/www` est le répertoire dans lequel se trouve la racine du serveur Web (dans notre exemple `htdocs`). Consultez, si nécessaire, le fichier de configuration de votre serveur Web afin de trouver le bon chemin.

Après avoir téléchargé l'archive dans le répertoire `/srv/www`, désarchivez-la à l'aide des commandes :

```
user@domain:/srv/www> tar xzf typo3_src-3.8.0.tar.gz
```

et

```
user@domain:/srv/www> tar xzf dummy-3.8.0.tar.gz
```

Ensuite, déplacez les dossiers dans le répertoire `dummy-3.8.0` à l'aide de la commande :

```
user@domain:/srv/www> mv dummy-3.8.0/* htdocs/
```


vers le répertoire `htdocs`, ou le répertoire qui contiendra votre site Web.
 Vous pouvez maintenant effacer les archives et le dossier vide.

```
user@domain:/srv/www> rm -r dummy-3.8.0
user@domain:/srv/www> rm dummy-3.8.0.tar.gz
```

Ensuite, si vous listez le contenu du répertoire `htdocs` avec la commande :

```
user@domain:/srv/www> ls -al htdocs/
```

vous devriez obtenir le résultat suivant :

```
total 38
drwxr-xr-x  6 user  group  512 May 23 02:42 .
drwxrwxr-x 14 user  group  512 Jul 24 17:51 ..
-rw-r--r--  1 user  group 4987 May 23 02:42 INSTALL.txt
-rw-r--r--  1 user  group  608 May 23 02:42 Package.txt
-rw-r--r--  1 user  group 8119 May 23 02:42 README.txt
-rw-r--r--  1 user  group  434 May 23 02:42 RELEASE_NOTES.txt
-rw-r--r--  1 user  group 4509 May 23 02:41 __.htaccess
-rw-r--r--  1 user  group   46 May 23 02:41 clear.gif
drwxr-xr-x  4 user  group  512 May 23 02:41 fileadmin
lrwxr-xr-x  1 user  group   18 Jul 24 17:51 index.php -> tslib/index_ts.php
lrwxr-xr-x  1 user  group   11 Jul 24 17:51 media -> tslib/media
lrwxr-xr-x  1 user  group   17 Jul 24 17:51 showpic.php -> tslib/showpic.php
lrwxr-xr-x  1 user  group   15 Jul 24 17:51 t3lib -> typo3_src/t3lib
lrwxr-xr-x  1 user  group   15 Jul 24 17:51 tslib -> typo3_src/tslib
lrwxr-xr-x  1 user  group   15 Jul 24 17:51 typo3 -> typo3_src/typo3
lrwxr-xr-x  1 user  group   18 Jul 24 17:51 typo3_src -> ../typo3_src-3.8.0
drwxr-xr-x  3 user  group  512 May 23 02:41 typo3conf
drwxr-xr-x  2 user  group  512 May 23 02:41 typo3temp
drwxr-xr-x  6 user  group  512 May 23 02:41 uploads
```

Exécutez les commandes suivantes afin de permettre au serveur Web d'accéder en écriture aux répertoires suivants :

```
chmod 777 typo3/temp
chmod 777 typo3/ext
chmod 777 typo3temp
chmod 777 typo3conf
chmod 777 typo3conf/ext
chmod 777 uploads
chmod 777 fileadmin
```

Donner les permissions `777` n'est pas sans risque, puisque cela donne tous les droits à l'ensemble des utilisateurs du serveur. Il serait préférable d'ajuster les permissions à `770`, si le propriétaire des répertoires peut être le Webmaster et si le groupe peut être celui sous lequel le serveur Web opère. Mais cela dépend des options d'administration et des permissions qui vous ont été accordées ; dans tous les cas, les permissions `777` fonctionnent. Ensuite, faites sauter la sécurité de l'outil d'installation en ouvrant le fichier suivant dans un éditeur (dans notre exemple `vi`, qui est disponible sur la plupart des plate-formes Linux).

```
user@domain:/srv/www> vi typo3/install/index.php
```

Au début, changez la ligne suivante :

```
die("In the main source distribution of TYPO3, the
install script is disabled by a die() function
call.<BR>Open the file typo3/install/index.php
and remove/out-comment the line that outputs this
message!");
```

Si vous utilisez vi (ou vim), tapez dd suivi de ZZ. Vous avez maintenant effacé la ligne, sauvegardé le fichier et quitté vi. Il est aussi possible de commenter la ligne en ajoutant // si plus tard vous souhaitez réactiver le verrouillage pour des raisons de sécurité. Il faut alors entrer i pour accéder au mode d'insertion. Tapez // au début de la ligne, quittez le mode d'insertion en appuyant sur (Esc), et entrez :wq! afin de sauvegarder vos changements et fermer l'éditeur.

Dans un navigateur, vous pouvez maintenant appeler l'outil d'installation, décrit au chapitre 2.4 à l'adresse :

```
http://www.votredomaine.com/typo3/install/
```

2.3.2 Installation WAMP

L'installation Windows de TYPO3 nécessite un système WAMP opérationnel et un serveur Apache avec les distributions les plus récentes de PHP et MySQL. Des programmes d'installation sont disponibles aux adresses URL <http://www.php.net/> ou <http://www.bigapache.org/>.

Après avoir installé Apache, MySQL et PHP, vous pouvez installer ImageMagick. Il s'agit aussi d'un logiciel libre ; vous trouverez à la référence ci-contre une version correspondante d'ImageMagick adaptée à l'utilisation de TYPO3. L'installation nécessite des exécutables si vous voulez éviter de compiler le logiciel par vous-même.

Référence 892286

Installez ImageMagick sur votre ordinateur avant de reprendre l'installation de TYPO3.

Téléchargez le paquetage dummy se trouvant dans la distribution zip de la page de téléchargement de TYPO3.org. Désarchivez le zip dans le répertoire source de votre site Web. Il s'agit généralement du répertoire suivant :

```
C:\Program Files\apache\htdocs\
```

Au cours de la prochaine étape, l'utilisateur sous lequel Apache opère aura besoin des droits de lecture et d'écriture sur les répertoires suivants :

```
typo3\temp\
typo3\ext\
typo3temp\
typo3conf\
typo3conf\ext\
uploads\
fileadmin\
```

Dans le dossier typo3\install\index.php, la ligne :

```
die("In the main source distribution of TYPO3, the install script is disabled by a die() function call.<BR>Open the file typo3/install/index.php and remove/out-comment the line that outputs this message!");
```

doit être effacée ou commentée par l'insertion de `//` en début de ligne.
Ensuite, passez à la section sur l'outil d'installation.

2.3.3 Installation WIIS

L'installation WIIS (Windows Installation Information Server) comporte sept étapes :

1. *Préparation du système*

Il est fortement recommandé de créer une partition séparée pour le serveur Web, afin d'éviter que les droits d'accès que vous devez configurer n'influencent pas les droits sur votre partition du système, ce qui mettrait votre système en danger.

2. *Installation de MySQL*

Téléchargez la dernière version de MySQL sur le site <http://www.mysql.com>, désarchivez le dossier d'installation et suivez les instructions. Lorsque l'installation est terminée, une fenêtre Windows s'ouvrira avec le programme WinMySQLadmin, dans laquelle vous pouvez définir les utilisateurs et les bases de données. Créez une base de données vide et un compte utilisateur pour cette base de données.

3. *Installation d'ImageMagick (optionnel)*

Référence 892286

ImageMagick est aussi un logiciel libre ; vous trouverez à la référence ci-contre une version correspondante d'ImageMagick adaptée à l'utilisation de TYPO3. L'installation nécessite des exécutables si vous voulez éviter de compiler le logiciel par vous-même.

4. *Installation de PHP*

Après l'installation de PHP, vous devez vérifier certains paramètres dans le fichier `php.ini` et les ajuster si nécessaire. Le fichier `php.ini` contient les paramètres de configuration PHP. Le paramètre le plus important pour le fonctionnement de TYPO3 est :

```
memory_limit=8M
```

Il devrait être augmenté à 16M (16 MB) minimum. L'outil d'installation vérifie quelques autres paramètres, mais ils sont en général déjà initialisés correctement dans la configuration par défaut.

5. *Configuration IIS*

La configuration IIS ne contient pas de données spéciales en ce qui concerne TYPO3 ; les réglages peuvent être soit pris en charge automatiquement par le programme d'installation PHP, soit ajustés manuellement, comme le décrit le guide d'installation de PHP. Pour des questions de performance, il est conseillé d'opérer PHP en mode ISAPI.

6. *Désarchivez TYPO3*

Désarchivez le paquetage ZIP sélectionné dans le répertoire cible, généralement appelé `F:\inetpub\wwwroot\`. Le nom de disque peut bien sûr varier.

7. *Assignment de droits d'accès NTFS*

Finalement, des droits doivent être donnés aux deux utilisateurs sous lesquels IIS est exécuté. Ces utilisateurs sont dénommés `IUSR` et `IWAM` suivis du nom de leur serveur. L'utilisateur `IUSR_NOMDEMACHINE` a besoin de droits de lecture sur tout le système de fichiers, afin d'exécuter les fonctions PHP `file_exists()`, `is_file()`, etc., et pas seulement pour les répertoires sous lesquels TYPO3 est installé. Cet utilisateur doit aussi

avoir des droits de lecture et d'écriture sur le programme `cmd.exe` afin d'utiliser Image-Magick là où il est nécessaire, ainsi que le droit de lecture sur le fichier `php.ini`. Dans le répertoire du serveur Web, les permissions suivantes doivent être allouées à l'utilisateur `IUSR_NOMDEMACHINE` :

TYPO3	Droit
Répertoire de serveur Web (généralement: Drive:\inetpub\wwwroot\)	Lire
fileadmin\	Écrire (répertoire et sous-répertoires)
typo3temp\	Écrire (répertoire et sous-répertoires)
uploads\	Écrire (répertoire et sous-répertoires)
typo3ext\	Écrire (répertoire et sous-répertoires)
typo3conf\	Écrire (répertoire et sous-répertoires)
C:\PHP\uploadtemp\	Écrire
ImageMagick	Lire et exécuter
C:\WINDOWS\system32\cmd.exe	Lire et exécuter

Tableau 2.1:
Droits de
`IUSR_NOMDEMACHINE`

2.4 L'outil d'installation

L'outil d'installation consiste essentiellement en une interface graphique qui permet d'éditer la configuration de TYPO3, sauvegardée dans le fichier `localconf.php` du répertoire `typo3conf/`. Le serveur Web doit donc avoir les droits d'écriture sur ce fichier et sur tout le répertoire `typo3conf`.

Essayons de comprendre le fonctionnement du système de configuration. À l'exécution, TYPO3 crée des fichiers tampon dans le répertoire `typo3conf` qui reprennent les paramètres de configuration. Ainsi, lorsque vous modifiez la configuration, ces fichiers tampon doivent être supprimés pour que les changements deviennent effectifs. La suppression se fait normalement automatiquement, sauf en cas de changement de version de code source de TYPO3, principalement vers une version inférieure ; vous devez alors supprimer manuellement les fichiers tampon. Ces fichiers ont des noms tels que `temp_CACHED_ps2268_ext_localconf.php`.

En ajoutant `/typo3/install/` au nom de votre domaine, vous appelez l'outil d'installation : `http://www.votredomaine.com/typo3/install/`.

Pour utiliser l'outil d'installation, il faut supprimer une fonction de verrouillage du script `typo3/install/index.php`, ce qui a déjà été décrit dans la section sur les installations LAMP et WAMP.

Le mot de passe par défaut pour l'outil d'installation est `joh316` ; vous devriez le changer immédiatement après votre première identification. Ensuite, vous démarrez l'installation en saisissant les informations suivantes :

1. nom d'utilisateur, mot de passe, nom d'hôte (localhost) et, si nécessaire, le nom d'une base de données déjà créée que TYPO3 devrait utiliser ;
2. le chemin menant au répertoire dans lequel est installé ImageMagick ; avec la commande

```
~$~locate identify
```

vous le découvrirez rapidement sur la plupart des distributions Linux.

L'outil d'installation est divisé en plusieurs sections dont les trois premières doivent être modifiées jusqu'à un certain point lors de l'installation. Les autres sections servent à la maintenance du système.

2.4.1 Basic Configuration

L'onglet de configuration de base vérifie les droits d'accès aux répertoires dans lesquels TYPO3 doit pouvoir écrire, et vérifie également la configuration PHP dans le fichier `php.ini`. Tous les problèmes pouvant empêcher l'installation sont indiqués ici avec les avertissements correspondants.

Figure 2.2:
Configuration
correcte pour laquelle
TYPO3 a les droits en
écriture sur les
répertoires de travail

Directories:	
✓	typo3temp/ writeable
✓	typo3conf/ writeable
✓	typo3conf/ext/ writeable
✓	typo3/ext/ writeable
✓	uploads/ writeable
✓	uploads/pics/ writeable
✓	uploads/media/ writeable
✓	uploads/tf/ writeable
✓	fileadmin/ writeable
✓	fileadmin/_temp_/ writeable

Vous devez aussi spécifier l'accès à votre base de données. Après avoir entré le nom d'utilisateur, le mot de passe et le nom d'hôte (généralement « localhost » si MySQL est installé sur votre serveur), vous soumettez le formulaire en cliquant sur **update configuration** soit pour sélectionner une base de données existante, soit pour en créer une nouvelle si vous en avez les droits (create).

Figure 2.3:
Accès à la base de
données

Username:	bt3entreprise
Password:	motdepasse
Host:	localhost

Nous poursuivons en passant en revue la configuration pour la création d'images. Si vous avez installé ImageMagick, spécifiez ici le chemin d'accès. TYPO3 recherche automatiquement ImageMagick dans le répertoire par défaut et détermine par ailleurs si vous avez compilé GDLib avec FreeType lors de l'installation PHP. Dans les versions récentes de FreeType, le texte dans l'image de test peut éventuellement déborder du cadre. Si ce problème persiste, nous le corrigerons lors d'une étape ultérieure (cf. section 2.4.4).

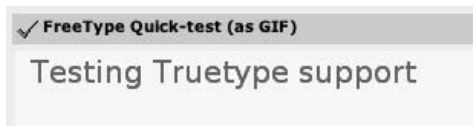


Figure 2.4:
Résultat du test
FreeType avec
résolution correcte

Lorsque vous soumettez le formulaire, l'outil d'installation sauvegarde toutes les valeurs des paramètres de configuration dans le fichier `localconf.php`.

2.4.2 Database Analyzer

Le script de l'analyse de base de données vous permet aussi bien d'éditer et de mettre à jour une base de données existante que d'en créer une nouvelle en spécifiant sa structure et son contenu. Une définition minimale de base de données est incluse dans le paquetage `dummy`. Le fichier SQL correspondant est dans le répertoire « » `typo3conf/` et est affiché automatiquement dans l'outil d'installation.

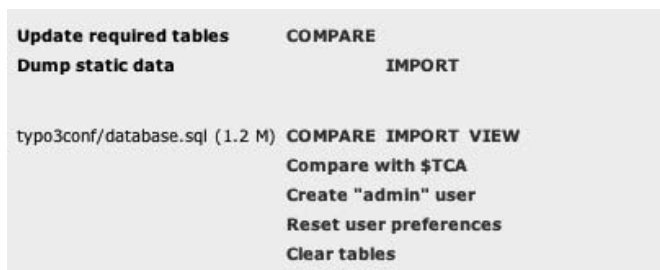


Figure 2.5:
Affichage du fichier
SQL par défaut
spécifiant la structure
et le contenu de la
base de données

Avec l'option **Import**, vous importez la base de données. Après avoir cliqué sur cette option, un message s'affiche, vous demandant de confirmer l'importation de toutes les données. Cochez l'option et soumettez le formulaire en cliquant sur **Write to database**.

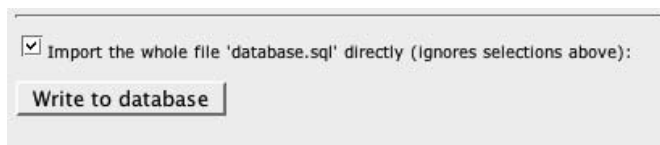


Figure 2.6:
Import de la base de
données

Cette opération peut prendre du temps selon les performances de votre serveur Web. À l'étape suivante, vous verrez une liste de toutes les tables de base de données accompagnées d'un

avertissement indiquant qu'elles existent déjà. Si ce n'est pas le cas, c'est-à-dire que certaines tables sont indiquées comme inexistantes, cela signifie que le processus d'import a échoué ; vous devriez donc réessayer. L'initialisation de la base de données est à présent terminée. Un compte administrateur est dès lors ouvert avec **admin** comme nom d'utilisateur et **password** comme mot de passe.

Figure 2.7:
Le message du
Database Analyzer
après un import
réussi du fichier SQL
(attention : d'autres
tables peuvent
s'afficher en fonction
de votre installation !



Pour initialiser la base de données, vous pouvez utiliser une méthode alternative qui ne réclame pas l'emploi du fichier **database.sql** du paquetage **dummy** – choisissez dans ce cas l'option **COMPARE**. Une liste de toutes les tables de base de données requises est indiquée. Celles-ci ont été présélectionnées. Appuyez sur le bouton de confirmation dans le formulaire pour créer les tables de base de données. Choisissez ensuite la fonction **Create admin user** et ouvrez un compte administrateur.

2.4.3 Image Processing

Ce script teste l'installation des bibliothèques de traitement d'images ImageMagick, de GDLib et Freetype. Nous n'en discutons pas en détail ici car il n'a pas d'influence sur la configuration de votre système.

2.4.4 All Configuration

Le script **All Configuration** contient des champs de saisie pour toutes les options de configuration de TYPO3 ainsi que quelques brèves explications de ces options dont les noms sont les clés du tableau PHP `TYPO3_CONF_VARS`. En général, seules quelques entrées sont importantes. Si le texte du test Freetype introduit précédemment dépasse du cadre, augmentez la résolution dans Freetype de 72 à 96 dpi.

Entrez cette valeur dans le champ `[GFX][TTFdpi]` ; vous le trouverez dans la partie finale de la section `[GFX]`, juste avant la section `[SYS]`.

Les autres entrées sont normalement optionnelles à ce stade.

2.4.5 typo3temp/

Cette section vous donne un ensemble de statistiques sur les fichiers enregistrés dans le répertoire `typo3temp/`, et vous permet aussi de les supprimer. Cette fonction est importante pour la maintenance de l'installation. Le répertoire est utilisé par TYPO3 pour enregistrer les fichiers d'images du frontend. Ces images sont régénérées chaque fois que le cache de la page est supprimé, tandis que les anciennes restent dans ce dossier. Pour les sites Web de grande envergure contenant une grande quantité d'images, plusieurs MBs peuvent rapidement s'y accumuler. Si les images sont supprimées de ce répertoire, bien que les pages pointent toujours vers les fichiers, l'image est régénérée pour autant que vous vidiez le cache de la page après avoir supprimé les fichiers. Vous pouvez également aller voir dans le backend de votre installation TYPO3, dans le module **Outils** → **Vérification BD** sous **Relations**, pour savoir combien de fichiers sans lien avec la base de données se trouvent dans ce répertoire.

2.4.6 phpinfo()

Cette section appelle la fonction PHP par défaut `phpinfo()` et affiche tous les paramètres essentiels de l'installation PHP. Vous trouverez tout au-dessus un résumé des paramètres les plus importants du système TYPO3, résumé que vous devriez insérer dans votre email en cas de question à la liste Install, de telle manière qu'une information complète soit fournie pour le débogage.

2.4.7 Edit files in typo3conf/

Cette section reprend la liste des fichiers situés dans le répertoire `typo3conf/`. Vous pouvez éditer chacun de ces fichiers ; cliquez simplement sur le nom du fichier pour lancer le formulaire d'édition. Cette option peut s'avérer très utile pour les utilisateurs de TYPO3 expérimentés, car elle permet d'insérer manuellement les valeurs de configuration, par exemple si l'un des scripts d'installation a échoué. Vous avez la possibilité de convertir tous les sauts à la ligne du format Windows au format Linux et de créer automatiquement une copie de sauvegarde du

fichier que vous éditez. Cette dernière fonction est hautement recommandée pour les essais de configuration dans le fichier `localconf.php`, afin que vous puissiez facilement restaurer la configuration précédente.

2.4.8 About

Vous connaissez déjà cette section : c'est la première page que vous apercevez lorsque le programme démarre. Elle vous permet de modifier le mot de passe de l'outil d'installation. Si vous ne l'avez pas encore fait, changez-le maintenant.

2.5 Options de configuration dans TYPO3_CONF_VARS

À côté de la configuration de base, un grand nombre de paramètres sont contrôlés par le script **All Configuration**. En voici la liste exhaustive, accompagnée pour chaque paramètre d'une description qui est parfois reprise dans l'outil d'installation sous une forme abrégée.

2.5.1 [GFX]:\$TYPO3_CONF_VARS["GFX"]

La zone GFX (pour « Graphics ») contient toutes les options de configuration pour le traitement d'images en TYPO3. IM est une abréviation pour ImageMagick et GD désigne la bibliothèque GD.

[image_processing]

Booléen (0,1). Active ou désactive le traitement d'images en TYPO3.

Exemple : `[image_processing] = 1`

[thumbnails]

Booléen (0,1). Affiche (ou non) les vignettes (images miniatures) dans le backend.

Exemple : `[thumbnails] = 1`

[thumbnails_png]

Bits. Bit 0 : l'entrée 0 crée des vignettes en format GIF, l'entrée 1 en format PNG. Bit 1 : l'entrée 2 spécifie que chaque fichier JPG est converti pour l'affichage de sa vignette en GIF ou en PNG avec l'entrée 3.

Exemple : `[thumbnails_png] = 0`

[nolconProc]

Booléen (0,1). Lorsque l'option est activée (1), les icônes ne sont pas construites dynamiquement dans le backend à partir de couches (par exemple, pour combiner l'icône de la page avec les icônes **Lancement**, **Arrêt** ou **Cacher la page**) mais doivent être disponibles sur le serveur. Cela peut être utile pour obtenir un backend opérationnel, même si les opérations de combinaison ne sont pas correctement supportées par le logiciel ImageMagick. Cette option devrait uniquement être désactivée si le serveur fournit toutes les fonctionnalités pour le traitement des images.

Exemple : `[nolconProc] = 1`

[gif_compress]

Booléen (0,1). Cette option active la fonction `t3lib_div::gif_compress()` qui recomprime

les fichiers GIF générés s'ils ne sont pas comprimés ou s'ils utilisent seulement la compression RLE (*Run Length Encoding*). Voir [im_path_lzw].

Exemple : [gif_compress] = 1

[imagefile_ext]

Vous spécifiez ici une liste d'extensions de fichiers (séparées par des virgules) qui devraient être interprétés comme des images par TYPO3. Lorsque IM n'est pas disponible, la liste doit être restreinte aux extensions gif,png,jpeg,jpg, écrites en minuscules et sans espace.

Exemple : [imagefile_ext] = gif,jpg,jpeg,tif,bmp,pcx,tga,png,pdf,ai

[gdlib]

Booléen (0,1). Cette option permet l'utilisation de la bibliothèque GD par TYPO3.

Exemple : [gdlib] = 1

[gdlib_png]

Booléen (0,1). Entrer 1 signifie que GD générera uniquement des fichiers PNG plutôt que des fichiers GIF. Cependant, les vieux navigateurs n'affichent pas les fichiers PNG. Et même les versions récentes de l'IE ne supportent pas toutes les fonctionnalités PNG comme, par exemple, les transparences.

Exemple : [gdlib_png] = 0

[gdlib_2]

Booléen (0,1). Vous devriez entrer la valeur 1 si votre serveur Web utilise GDLib 2.0.1 ou plus, sans quoi certains problèmes pourraient survenir.

Exemple : [gdlib_2] = 0

[im]

Booléen (0,1). Permet l'utilisation d'ImageMagick (IM) par TYPO3.

Exemple : [im] = 1

[im_path]

Entrez ici le chemin menant au dossier dans lequel les programmes IM convert, combine et identify sont situés sur le serveur Web.

Exemple : [im_path] = /usr/local/bin/

[im_path_lzw]

Entrez les détails du chemin pour la version IM dont la commande convert peut implémenter la compression LZW. Cette compression a été brevetée par Unisys et fut supportée temporairement par ImageMagick. La version recommandée 4.2.9 d'IM peut être équipée avec la compression LZW ; dans ce cas le champ est laissé vide et [gif_compress] est désactivé. On peut attendre des versions futures de plusieurs logiciels qu'elles supportent la compression LZW, étant donné que le brevet a expiré.

Exemple : [im_path_lzw] = /usr/local/typo3sh/bin/

[im_version_5]

Booléen (0,1). Mettez cette valeur à 1 si vous utilisez une version d'IM 5.x ou plus.

Exemple : [im_version_5] = 0

[im_negate_mask]

Booléen (0,1). À partir de la version 5.1, les images doivent être insérées avant d'être combinées avec un masque.

Exemple : `[im_negate_mask] = 0`

[im_imvMaskState]

Booléen (0,1). Depuis la version 5.4.3+, la situation décrite dans le paragraphe précédent a été inversée et TYPO3 peut être configuré de telle manière que le réglage global `[im_version_5]` ne demande pas la conversion des fichiers d'image qui doivent être masqués, ce qui était nécessaire auparavant.¹⁰

Exemple : `[im_imvMaskState] = 0`

[im_no_effects]

Booléen (0,1). Avec la version 4.2.9 recommandée, les effets d'ImageMagick fonctionnent bien. Dans les versions plus récentes, ils sont devenus plus lents, et peuvent être désactivés à l'aide de ce paramètre. Enfin, dans les versions d'IM les plus récentes, les effets fonctionnent bien mieux mais la syntaxe de l'API a une nouvelle fois été modifiée – sans commentaires ...

Exemple : `[im_no_effects] = 0`

[im_v5effects]

Entier (-1,0,1). 0 = désactivé. -1 = ne pas rendre les contours des images plus nets par défaut. 1 = Tous ; la définition des contours (sharpening) et le flou (blurring) sont activés et l'option `[im_no_effects]` est annulée.

Exemple : `[im_v5effects] = 0`

[im_mask_temp_ext_gif]

Booléen (0,1). Activez cette option si vous utilisez une version d'IM 5+. La classe par défaut `tslib_cObj` utilise le format PNG car la génération est plus rapide et demande moins de CPU. Etant donné que certaines versions d'IM au-delà de la version 5 ne supportent pas le format PNG correctement, cela peut être supprimé en activant cette option.

Exemple : `[im_mask_temp_ext_gif] = 0`

[im_mask_temp_ext_noloss]

Chaîne de caractères. Pendant que les images sont masquées, il faut bien entendu enregistrer les fichiers temporaires dans un format sans perte. Pour ce faire, le format `miff` d'ImageMagick est idéal. Malheureusement, la version 5.4.9 d'ImageMagick n'est pas capable de générer son propre format de fichier. Dans ce cas, en cas de problème avec les masques, vous pouvez utiliser au choix les formats TIF, PNG, or JPG.

Exemple : `[im_mask_temp_ext_noloss] = miff`

[im_noScaleUp]

Booléen (0,1). Les images ne sont pas agrandies lorsque l'option est activée.

Exemple : `[im_noScaleUp] = 0`

¹⁰Ceci a mené, de pair avec beaucoup d'autres développements, à la séparation du projet GraphicsMagic afin de créer une interface plus fiable. Le commentaire de Kasper Skårhøj à ce moment venait clairement du cœur : « Alléluia pour ImageMagick – ai-je un jour regretté d'utiliser ce paquetage... »

[im_combine_filename]

Chaîne de caractères. Les versions d'IM les plus récentes ont renommé la commande **combine** en **composite** ; donnez ici le nom correct.

Exemple : `[im_combine_filename] = combine`

[im_noFramePrepended]

Booléen (0,1). Certains formats d'image comme GIF ou TIF permettent de sauver plusieurs images dans un seul fichier. ImageMagick fournit une option pour travailler uniquement avec la première image, ce qui accroît généralement la vitesse de travail. Malheureusement, quelques versions d'IM contiennent une erreur impliquant qu'IM ignore tout bonnement ces images. Le cas échéant, cette option doit être activée.

Exemple : `[im_noFramePrepended] = 0`

[enable_typo3temp_db_tracking]

Booléen(0,1). Vous spécifiez ici si tous les fichiers dans **typo3temp/** doivent être enregistrés dans une table de la base de données. Ceci empêche les images d'être créées simultanément par deux processus différents, puisque la relation entre le fichier de sortie temporaire et le fichier source est notée ici. En outre, le module **Outils** → **Vérification BD** du backend et l'outil d'installation vous renseigne sur le nombre de vieux fichiers se trouvant dans le répertoire **temp/**.

Exemple : `[enable_typo3temp_db_tracking] = 0`

[TTFLocaleConv]

Chaîne de caractères. Vous pouviez spécifier ici, jusqu'à la version 3.6.0, le format de sortie des fonctions TrueType. Depuis la version 3.6.0, la sortie est toujours de type UTF-8.

Exemple : `[TTFLocaleConv] =`

[TTFdpi]

Entier. Cette option importante permet de régler la résolution en dpi (pixels par pouce) dans laquelle fonctionne le système Freetype sur votre serveur. Cette résolution est de 96 dpi depuis la version 2 pour 72 avant ; l'affichage des polices est alors trop grand si la valeur n'est pas changée à 96 dpi pour les versions plus récentes.

Exemple : `[TTFdpi] = 96`

[im_jpg_quality]

Entier. Cette valeur détermine la qualité du format JPEG.

Exemple : `[im_jpg_quality] = 70`

2.5.2 [SYS]:\$TYPO3_CONF_VARS["SYS"]

Cette section décrit les options de configuration des interfaces backend et frontend.

[textfile_ext]

Indiquez ici, via les extensions de fichiers, quels types de fichiers peuvent être édités dans le backend.

Exemple : `[textfile_ext] = txt,html,htm,css,inc,php,php3,tmpl,js,sql`

[contentTable]

Cette option vous permet de spécifier le nom de la table des éléments de contenu de la page. La valeur par défaut est `tt_content`.

Exemple : `[contentTable] = tt_content`

[sitename]

Il s'agit du nom de l'installation qui est montré au sommet de l'arbre de la page, à côté de l'icône représentant le globe terrestre. Le nom est aussi disponible dans la section **Basic Configuration** de l'outil d'installation.

Exemple : `[sitename] = BT3-Entreprise`

[ddmmyy]

Le format d'affichage de la date — correspond à la notation de la fonction `date()` en PHP.

Exemple : `[ddmmyy] = d.m.y`

[hhmm]

Le format d'affichage de l'heure — correspond à la notation de la fonction `date()` en PHP.

Exemple : `[hhmm] = H:i`

[encryptionKey]

Cette option permet de spécifier une chaîne de caractères aléatoire utilisée dans la création de valeurs de hachage pour le chiffrement dans le menu contextuel, le **Direct Mail Module**, ainsi qu'à d'autres endroits dans le système ; cela permet d'améliorer la sécurité.

Exemple : `[encryptionKey] = Haaken Flip`

[doNotCheckReferer]

Booléen (0,1). Cette option permet de désactiver le contrôle actif dans le backend, qui vérifie que l'hôte accédant et l'hôte référent (*referring host*) sont bien identiques. Si la valeur est mise à 1, la vérification n'est plus active. Cela peut être utile lorsque l'accès se fait via des serveurs proxy qui ne donnent pas une valeur correcte à la variable `HTTP_REFERER`.

Exemple : `[doNotCheckReferer] = 0`

[recursiveDomainSearch]

Booléen (0,1). Lorsque cette option est activée (1), si l'on essaie d'accéder à un domaine non-existant, TYPO3 efface récursivement des parties du nom jusqu'au moment où il trouve une correspondance avec un nom de domaine configuré dans TYPO3.

Exemple : `[recursiveDomainSearch] = 0`

[T3instID]

Cette option n'a pas encore été utilisée. L'intention était de créer une identité unique

à laquelle chaque installation peut s'identifier lorsqu'elle accède au répertoire d'extensions. Cela servirait à des fins statistiques, mais n'a pas encore été mis en pratique.

Exemple : [T3instID] = N/A

[devIPmask]

Indiquez les adresses IP séparées par des virgules. Cette option importante définit la liste des adresses IP pour lesquelles les messages d'erreur sont affichés dans le frontend. La fonction `Debug()` utilise ces entrées comme filtre. Une entrée vide ne permet aucun accès. Le caractère `*` donne accès à tous les hôtes. Ce même caractère (`*`) permet de donner l'accès à plusieurs adresses IP ayant certains chiffres en commun. Par exemple, `192.168.*.*` donne l'accès à toutes les adresses IP commençant par `192.168.`.

Exemple : [devIPmask] = 192.168.*,127.0.0.1

[curlUse]

Booléen (0,1). En entrant 1, la fonction `getUrl` utilisée par le système est `curl` au lieu de `fopen()`, de telle manière que vous puissiez travailler avec des serveurs proxy (ce qui n'est pas possible avec `fopen()`). La bibliothèque Curl est certainement disponible dans votre installation PHP.

Exemple : [SYS][curlUse] = 0

[curlProxyServer]

URL. Vous devez donner l'adresse de votre serveur proxy à Curl sous la forme `http://proxy:port/`.

Exemple : [curlProxyServer] = http://192.168.1.1:8080

[curlProxyTunnel]

Booléen (0,1). Pour des raisons de sécurité, il est nécessaire d'établir un tunnel à travers le serveur proxy. Entrez 1 et Curl s'en chargera.

Exemple : [curlProxyTunnel] = 0

[curlProxyUserPass]

Chaîne de caractères. Vous entrez vos nom d'utilisateur et mot de passe pour l'accès au serveur proxy, si nécessaire, en utilisant la notation `Nom d'utilisateur:Mot de passe`.

Exemple : [curlProxyUserPass] = Leeloo:Multipass

[form_ctype]

Chaîne de caractères. Cette option permet d'ajuster globalement le type de chiffrement de la plupart des formulaires dans TYPO3. `multipart/form-data` est l'option par défaut ; elle permet de charger les fichiers. Si le chargement de fichiers n'est pas permis par votre installation PHP, les données créées avec ce formulaire ne seront pas transférées. Le type de chiffrement peut être changé en conséquence par la valeur `application/x-www-form-urlencoded`.

Exemple : [form_ctype] = multipart/form-data

[loginCopyrightWarrantyProvider]

Chaîne de caractères. GPL n'inclut aucune garantie de la part de l'auteur du logiciel. Si vous désirez ou devez assumer la garantie de la fonction envers un client, vous pouvez entrer votre nom dans ce champ de sorte qu'il soit affiché sur la page d'ouverture de session. Une adresse Internet (URL) est aussi donnée à l'étape suivante.

Exemple : [loginCopyrightWarrantyProvider] = VEB Optimismus

[loginCopyrightWarrantyURL]

Chaîne de caractères. Spécifiez une adresse URL de la forme `http://www.votredomaine.com` ; elle servira de lien aux noms donnés dans l'option précédente.

Exemple : [loginCopyrightWarrantyURL] = `http://www.veb-optimismus.de`

[loginCopyrightShowVersion]

Booléen (0,1). La page d'ouverture de session indiquera la version TYPO3 si vous entrez la valeur 1.

Exemple : [loginCopyrightShowVersion] = 0

[binPath]

Chaîne de caractères sous forme de liste séparée par des virgules. Vous entrez une liste de chemins absolus dans lesquels la recherche de programmes externes est faite. Ceci est par exemple utilisé par les services de l'extension DAM.

Exemple : [binPath] = `/usr/bin/`

[t3lib_cs_convMethod]

Chaîne de caractères. Dans la classe `t3lib_cs`, entrer l'une des valeurs suivantes spécifie avec quel outil les jeux de caractères sont convertis : `iconv` ou `recode` sont des programmes externes ; l'argument par défaut est le code PHP propre à TYPO3. Les programmes externes sont significativement plus rapides mais l'encodage HTML de caractères spéciaux n'est pas supporté.

Exemple : [t3lib_cs_convMethod] = `recode`

[t3lib_cs_utils]

Chaîne de caractères. Au lieu d'utiliser le code propre à TYPO3 pour convertir les jeux de caractères, vous pourriez vouloir utiliser le module PHP `mbstring` qui est bien plus rapide. Pour ce faire, il vous suffit d'entrer le nom de ce module en argument. La chaîne `mbstring` est d'ailleurs la seule qu'accepte cette option comme argument, contrairement à ce qui est indiqué dans la description anglaise de l'outil d'installation. Si vous n'entrez rien, le code propre à TYPO3 sera utilisé.

Exemple : [t3lib_cs_utils] = `mbstring`

[enable_DLOG]

Booléen. Indique si le log « développeur » est activé. Voir la constante `TYPO3_DLOG`.

Exemple : [enable_DLOG] =

[no_pconnect]

Booléen. La valeur 1 force l'utilisation de `connect` à la base de données à la place de `pconnect`.

Exemple : `[no_pconnect] = 1`

[multiplyDBfieldSize]

Valeur décimale comprise entre 1 et 5. Indique un facteur par lequel sera multipliée la taille de chaque champ lorsque l'outil d'installation calcule la taille de la base de données (p.ex.2,5). C'est utile lorsque vous voulez augmenter la taille des champs pour utf-8. Pour des sites de l'Europe de l'Ouest utilisant utf-8, on ne devrait pas avoir une valeur supérieure à deux fois la taille normale d'un octet simple et pour les langues asiatiques, un facteur 3 devrait suffire.

Exemple : `[multiplyDBfieldSize] = 1`

[setMemoryLimit]

Entier. Indique la valeur du paramètre PHP `memory_limit` en MB : si elle est supérieure à 16, TYPO3 essaie d'utiliser `ini_set()` pour fixer la limite mémoire de PHP à cette valeur. Cette opération n'a de sens que si votre sysadmin vous a donné les droits d'utiliser la fonction `ini_set()`.

Exemple : `[setMemoryLimit] = 0`

[forceReturnPath]

Booléen. Force l'utilisation du chemin de retour (return path) dans la fonction `mail()`. Si cette valeur est active, tous les appels à `mail()` par la classe `t3lib_htmlmail` seront faits avec `-f` comme cinquième paramètre. Cela rendra le chemin de retour correct sur la plupart des systèmes UNIX. Il y a un problème avec les versions inférieures à 2 pour Postfix : les emails ne sont pas envoyés si cette valeur est activée. Sur les plate-formes Windows, le chemin de retour est donné via un appel à `ini_set`. Cela n'a pas d'effet si PHP est installé en `safe_mode`.

Exemple : `[forceReturnPath] = 0`

[displayErrors]

Entier, -1,0,1. 0=N'afficher aucun message d'erreur PHP. 1=Afficher les messages d'erreur. -1=Valeur par défaut qui permet de remplacer le paramètre `display_errors`. Il est conseillé de fixer la valeur de ce paramètre à 0 et d'activer à la place l'option `error_log` dans `php.ini`.

Exemple : `[displayErrors] = 1`

[serverTimeZone]

Entier. Spécifie le décalage en heures par rapport à l'heure GMT. La valeur par défaut est 1 qui correspond à l'heure GMT+1 (Europe centrale). Cette valeur peut être utilisée dans les extensions qui doivent convertir des heures à partir ou vers d'autres fuseaux horaires.

Exemple : `[serverTimeZone] = 1`

2.5.3 [EXT]:\$TYPO3_CONF_VARS["EXT"]

Nous décrivons dans cette section différentes options pour configurer le gestionnaire d'extensions et les parties du système s'y rapportant.

[noEdit]

Booléen (0,1). La valeur 1 empêche les fichiers dans le gestionnaire d'extensions d'être édités. Pour les développeurs, la valeur devrait être mise à 0 afin qu'ils puissent y apporter directement les changements.

Exemple : [noEdit] = 1

[allowGlobalInstall]

Booléen (0,1). La valeur 1 permet l'installation des extensions globales dans le répertoire `typo3/ext/` via le gestionnaire d'extensions, ainsi que leur mise à jour et leur suppression dans ce répertoire. Cette option est importante pour que les administrateurs ne puissent pas modifier les extensions globales au cas où plusieurs sites Web sont installés sur le serveur.

Exemple : [allowGlobalInstall] = 0

[allowLocalInstall]

Booléen (0,1). La valeur 1 permet d'administrer les extensions locales dans le répertoire `typo3conf/ext/` de l'instance en question via le gestionnaire d'extensions.

Exemple : [allowLocalInstall] = 1

[em_devVerUpdate]

Booléen (0,1). La valeur 1 provoque un marquage rouge des versions des extensions si des mises à jour sont disponibles.

Exemple : [em_devVerUpdate] = 0

[em_alwaysGetOOManual]

Booléen (0,1). Vous demandez, en entrant la valeur 1, que la documentation incluse dans les extensions, disponible en format OpenOffice, soit toujours téléchargée depuis le serveur.

Exemple : [em_alwaysGetOOManual] = 0

[em_systemInstall]

Booléen (0,1). Vous permettez au gestionnaire d'extensions d'installer les extensions du répertoire `sysext/` en entrant la valeur 1. C'est nécessaire pour réaliser les mises à jour des extensions des systèmes `cms` et `lang` à partir du gestionnaire d'extensions.

Exemple : [em_systemInstall] = 0

[requiredExt]

Chaîne de caractères sous forme de liste séparée par des virgules. Vous spécifiez les extensions qui ne peuvent être désactivées par le Gestionnaire d'Extensions (GE). Normalement, les extensions `cms` et `lang` qui forment le cœur du système sont incluses dans la liste.

Exemple : [requiredExt] = cms,lang

[extCache]

Entier (0,1,2,3). Pour ne pas enregistrer les scripts d'extensions `ext_local-conf.php` et `ext_tables.php` dans le cache, entrez 0. Les scripts seront alors lus chaque fois à chaque requête de la page —ce qui est intéressant lorsque vous développez les extensions, puisque cela vous permet de ne pas devoir supprimer le cache dans le backend.

L'argument par défaut est 1. Les scripts sont saués dans des fichiers sous la forme `typo3conf/temp_CACHED_[sitePathHash]*`, ce qui réduit la charge du serveur et augmente ses performances —un réglage utile pour un serveur en production.

La valeur 2 fait en sorte que le nom de fichier du fichier tampon contienne une valeur de hachage basée sur la chaîne de caractères `[extList]`. La valeur 3 signifie que les noms des fichiers tampon ne contiendront aucune valeur de hachage.

Exemple : `[extCache] = 1`

[extList]

Chaîne de caractères sous forme de liste séparée par des virgules. Les extensions installées par le gestionnaire d'extensions sont entrées dans cette liste. Si vous installez une extension défectueuse qui rend impossible le travail avec le système, éditez le fichier `typo3conf/localconf.php` pour enlever cette extension et effacer la clé de cette liste. En outre, les fichiers `typo3conf/temp_CACHED_*` doivent être supprimés pour que le système soit à nouveau opérationnel.

Exemple : `[extList] = tsconfig_help,context_help,extra_page_cm_options,...`

2.5.4 [BE]:\$TYPO3_CONF_VARS["BE"]

Cette section présente les paramètres de la configuration backend.

[unzip_path]

Indiquez le chemin menant au programme de décompression `unzip`.

Exemple : `[unzip_path] = /usr/bin/`

[diff_path]

Indiquez le chemin vers l'application `diff` en ligne de commande, utilisée pour comparer les fichiers. Une version Windows de `diff` peut se télécharger à l'adresse :

<http://unxutils.sourceforge.net/>

Exemple : `[diff_path] = diff`

[fileadminDir]

Le chemin du répertoire `fileadmin` ; chemin relatif par rapport au chemin du site spécifié par la constante `PATH_site`.

Exemple : `[fileadminDir] = fileadmin/`

[RTE_imageStorageDir]

Le chemin vers le répertoire dans lequel seront saués les fichiers du Rich Text Editor.

Exemple : `[RTE_imageStorageDir] = uploads/`

[staticFileEditPath]

Le chemin vers le répertoire dans lequel sont enregistrés et édités les fichiers « statiques ». Les champs de la base de données peuvent être configurés dans le \$TCA de telle sorte qu'ils sont en réalité stockés dans un fichier. L'extension `sys_staticfile_edit` en est une application.

Exemple : `[staticFileEditPath] = fileadmin/static/`

[lockRootPath]

Chaîne de caractères. Spécifie la partie commune des chemins `[userHomePath]` et `[groupHomePath]`. Notez que la première partie des chemins `[userHomePath]` et `[groupHomePath]` doit correspondre à `[lockRootPath]`. Cette valeur est aussi utilisée pour accéder à un répertoire en dehors des valeurs `PATH_site`. Par exemple, on change cette valeur si un traitement sur des données doit être autorisé à un niveau au-dessus du répertoire racine du serveur Web.

Exemple : `[lockRootPath] = /srv/www/archiv/`

[userHomePath]

Chaîne de caractères. Chemin vers le répertoire personnel de l'utilisateur backend. Si la valeur du chemin est insérée ici, TYPO3 crée automatiquement un répertoire personnel pour chaque utilisateur. Par exemple, si la valeur est `/home/typo3/users`, alors un répertoire avec le chemin `/home/typo3/users/43_cameronfrye/` est créé pour l'utilisateur `43_cameronfrye` avec l'uid 43.

Exemple : `[userHomePath] = /srv/www/htdocs/typo3/users/`

[groupHomePath]

Chaîne de caractères. Comme pour les utilisateurs, un répertoire séparé est créé automatiquement pour chaque groupe.

Exemple : `[groupHomePath] = /srv/www/htdocs/typo3/group/`

[userUploadDir]

Chaîne de caractères. Un suffixe préconfigurable ajouté aux répertoires de l'utilisateur. Si le répertoire de l'utilisateur correspond au nom d'utilisateur et à l'uid, c'est-à-dire ici `43_cameronfrye/` et que la valeur spécifiée est `/250GT`, alors un répertoire nommé `43_cameronfrye/250GT/` est mis à disposition de l'utilisateur.

Exemple : `[userUploadDir] = /250GT`

[fileCreateMask]

Définissez ici les permissions d'accès données aux fichiers créés par TYPO3 dans le système de fichiers, en accord avec la syntaxe de la commande `umask` de UNIX.

Exemple : `[fileCreateMask] = 0644`

[folderCreateMask]

Le même réglage que le précédent, mais cette fois pour les dossiers.

Exemple : `[folderCreateMask] = 0755`

[warning_email_addr]

Une adresse email à laquelle est envoyé un avertissement lorsque quatre tentatives manquées d'ouverture de session au backend se sont produites en une heure.

Exemple : [warning_email_addr] = ronald@evilempire.com

[warning_mode]

Entier (1,2). Si vous entrez 1, un message est envoyé à l'adresse donnée précédemment chaque fois qu'un utilisateur a ouvert une session au backend. Si vous entrez 2, l'adresse reçoit un avertissement uniquement lorsqu'un administrateur ouvre une session.

Exemple : [warning_mode] = 2

[IpmaskList]

Chaîne de caractères. Spécifiez les adresses IP auxquelles vous octroyez exclusivement un accès au backend. Les utilisateurs ayant d'autres adresses IP n'auront donc aucun accès. Il est possible d'utiliser * comme caractère *joker*¹¹.

Exemple : [IPmaskList] = 192.168.1.*

[adminOnly]

Booléen (0,1). En spécifiant 1, seuls les administrateurs peuvent accéder au backend. La valeur 0 donne l'accès à tous les utilisateurs. Cette option peut être utile pour exclure les utilisateurs du système lors des travaux de maintenance et de mise à jour.

Exemple : [adminOnly] = 0

[lockBeUserToDBmounts]

Booléen (0,1). La valeur par défaut 1 octroie uniquement aux utilisateurs l'accès à leur propre Pagemount, ce qui peut être désactivé en entrant 0. Ce dernier scénario est hautement improbable.

Exemple : [lockBeUserToDBmounts] = 1

[lockSSL]

Entier (0,1,2). Les valeurs 0 et 1 demandent à TYPO3 de rendre le backend disponible uniquement via une connexion SSL. La valeur 2 fait en sorte que l'accès à <http://votredomaine.com/typo3> soit automatiquement redirigé vers <https://votredomaine.com/typo3>.

Exemple : [lockSSL] = 0

[disable_exec_function]

Booléen (0,1). Vous supprimez l'utilisation de la fonction PHP `exec()` en entrant 1, ce qui peut être utile sous Windows. Pour ImageMagick, la même action se fait en désactivant toutes les fonctions graphiques : [GFX][im]=0.

Exemple : [disable_exec_function] = 0

¹¹NdT : *wildcard* en anglais

[usePHPFileFunctions]

Booléen (0,1). Quand PHP opère en mode `safe_mode`, toutes les opérations sur les fichiers doivent être passées par le biais de fonctions PHP plutôt que par des appels à des commandes externes via la fonction `exec()`. C'est le comportement adopté lorsque la valeur est mise à 1.

Exemple : `[usePHPFileFunctions] = 1`

[compressionLevel]

Entier (1-9). Requiert zlib en PHP. La compression avec `gzip` se fait en entrant une valeur de 1 à 9. Les pages comprimées utilisent moins de bande passante mais la charge du CPU augmente avec des taux de compression plus importants. L'entrée 0 ne réalise aucune compression, l'entrée 9 produit une compression maximale. Alternativement, si vous spécifiez `TRUE`, la compression est ajustée dynamiquement selon la charge du système (seulement sous Linux et FreeBSD). La compression peut être également configurée dans Apache.

Exemple : `[compressionLevel] = 0`

[MaxFileSize]

Entier. Indiquez la taille maximale des fichiers édité par TYPO3. Cette valeur est pertinente uniquement dans le contexte des tailles de fichiers définis pour PHP dans le fichier `PHP.ini`.

Exemple : `[maxFileSize] = 10000`

[RTEenabled]

Booléen(0, 1). Cette option vous permet d'activer (1) ou de désactiver (0) globalement le Rich Text Editor indépendamment de la configuration dans le backend.

Exemple : `[RTEenabled] = 1`

[forceCharset]

Chaîne de caractères. Le jeu de caractères est normalement en accord avec le jeu de langues dans le backend des utilisateurs respectifs. Cette option permet de le définir pour tous les utilisateurs. Les options sont indiquées dans les tables de jeu de caractères et dans le répertoire `t3lib/csconvtbl/`. Pour l'encodage Unicode par exemple, vous pouvez utiliser `utf-8`. Spécifiez les jeux de caractères en minuscules.

Exemple : `[forceCharset] = iso-8859-8`

[installToolPassword]

Chaîne de caractères. Il s'agit de la valeur de hachage du mot de passe pour l'outil d'installation. Pour bloquer l'accès, n'entrez rien. Il est judicieux de protéger en outre le répertoire de l'outil d'installation, `typo3/install/`, à l'aide d'un mot de passe via un fichier `.htaccess`.

Exemple : `[installToolPassword] = e1c102cf0300bf73e47018f5bd7766e5`

[trackBeUser]

Booléen (0,1). Entrez la valeur 1 pour demander à TYPO3 de tracer dans la table `sys_`

trackbeuser chaque appel d'un script dans le backend. Vous devez aussi installer l'extension `beuser_tracking`.

Exemple : `[trackBeUser] = 0`

[defaultUserTSconfig]

Chaîne de caractères. TypeScript : cette option permet de définir du code TypeScript par défaut valable pour tous les utilisateurs backend (voir chapitre 4.8).

Exemple : `[defaultUserTSconfig] = admPanel.enable= 1`

[defaultPageTSconfig]

Chaîne de caractères. TypeScript : cette option permet de définir du code TypeScript par défaut valable pour toutes les pages (voir chapitre 4.8).

Exemple : `[defaultPageTSconfig] = mod.web_layout.tt_content.colPos_list = 0,3`

[enabledBeUserIPlock]

Booléen (0,1). En entrant 1, vous activez l'option User/Group TSConfig `option.lockToIP`. Une configuration plus poussée est possible via le TSConfig de l'utilisateur ou du groupe à paramétrer.

Exemple : `[enabledBeUserIPlock] = 1`

[fileDenyPattern]

Chaîne de caractères. Spécifiez les séquences de caractères selon la syntaxe de la fonction `eregi()`. Les fichiers correspondant à ces séquences ne sont pas renommés ou chargés sur le serveur.

Exemple : `[fileDenyPattern] = \.php\.\.php3\.`

[interfaces]

Indiquez ici à quelles interfaces l'utilisateur accède après avoir ouvert une session dans le backend, et dans quel ordre. Les choix possibles sont `backend` et `frontend` (séparés par une virgule le cas échéant).

Exemple : `[interfaces] = backend`

[loginLabels]

Les options d'entrée de l'écran d'ouverture de session sont réécrites avec d'autres expressions, en français par exemple.

Exemple : `[loginLabels] = Utilisateur|Mot de passe|Interface|Ouvrir une session|Fermer la session|...`

[notificationPrefix]

Cette option permet de créer un en-tête pour les messages du système à l'administrateur.

Exemple : `[notificationPrefix] = Les miracles s'accomplissent parfois...`

[createGroup]

Entier. Spécifie le groupe lors de la création de nouveaux fichiers et répertoires. Le

groupe peut être changé sur les système UNIX. Activez cette option si vous voulez changer le groupe des fichiers et des répertoires. C'est utile dans tous les cas où le serveur Web opère avec un utilisateur ou un groupe différent du vôtre. Dans ce cas, créez un nouveau groupe, et ajoutez-vous-y, ainsi que l'utilisateur du serveur Web. Dès ce moment, vous pouvez fixer le dernier bit de `fileCreateMask/folderCreateMask` à 0 (p.ex. 770). Important : l'utilisateur sous lequel votre serveur Web opère doit être un membre du groupe que vous spécifiez ici. Sinon, vous risquez de voir surgir des erreurs.

Exemple : `[createGroup]=`

`[lockIP]`

Entier (0-4). Verrouillage de la session IP des utilisateurs backend. Voir `[FE][lockIP]` pour plus de détails. La valeur par défaut est 4.

Exemple : `[lockIP]=4`

`[sessionTimeout]`

Entier. Spécifie la durée maximale d'une session pour les utilisateurs backend. La valeur par défaut est 3600 secondes, soit une heure.

Exemple : `[sessionTimeout]=3600`

`[loginSecurityLevel]`

Chaîne de caractères. Mot-clé qui détermine le niveau de sécurité du login dans le backend. **normal** signifie que le mot de passe du formulaire d'identification est envoyé « en clair ». **challenge** veut dire que le password n'est pas envoyé, mais qu'une valeur de hachage est calculée. En spécifiant **superchallenged**, la valeur de hachage est calculée sur le mot de passe avant que cette valeur soit elle-même combinée avec la valeur **challenge** (de cette manière, la valeur de hachage du mot de passe est enregistrée dans la base de données et non plus le mot de passe lui-même). NE PAS CHANGER cette valeur manuellement ; sans un autre service d'identification, cela empêche les ouvertures de session dans TYPO3 puisque la méthode « **superchallenged** » est encodée dans le système d'identification par défaut.

Exemple : `[loginSecurityLevel]=normal`

`[useOnContextMenuHandler]`

Booléen. Si l'option est activée, le clic droit (right-click) du menu contextuel est activé dans le backend — même si ce n'est pas un attribut XHTML !

Exemple : `[useOnContextMenuHandler]=1`

`[accessListRenderMode]`

`[explicitADmode]`

2.5.5 `[FE]:$TYPO3_CONF_VARS["FE"]`

Les paramètres de configuration dans la section suivante se réfèrent au frontend, c'est-à-dire aux sites Web publiés par TYPO3.

[png_to_gif]

Booléen (0,1). La valeur 1 active la conversion de tous les fichiers PNG générés dans le frontend en fichiers GIF, ce qui laisse un grand nombre de fichiers temporaires dans le répertoire `typo3temp/`.

Exemple : `[png_to_gif] = 0`

[tidy]

Booléen (0,1). Si vous indiquez 1, le code HTML est nettoyé et optimisé à l'aide du programme `tidy`. Cette option est recommandée, surtout pendant les périodes de développement, de manière à ce que le code HTML généré soit plus lisible. Souvenez-vous toutefois que `tidy`, selon les options, nettoie ou répare le code HTML défectueux. Il est préférable de désactiver cette option pour les systèmes en ligne pour éviter de charger inutilement le serveur. `tidy` est disponible à l'adresse suivante : <http://www.w3.org/People/Raggett/tidy/>.

Exemple : `[tidy] = 0`

[tidy_option]

Options : `[all, cached, output]`. `all` provoque le nettoyage par `tidy` de tout le contenu avant qu'il soit sauvé, ou pas, dans le cache. `cached` nettoie le contenu uniquement avant qu'il soit sauvé dans le cache. `output` nettoie le code HTML seulement s'il est demandé à partir du cache.

Exemple : `[tidy_option] = cached`

[tidy_path]

Spécifiez la commande `tidy`, chemin et options inclus, à l'endroit adéquat. D'autres paramètres que ceux par défaut peuvent être définis suivant la documentation de `tidy`. Pour générer du XHTML par `tidy`, ajoutez l'expression `-output-xhtml true`.

Exemple : `[tidy_path] = tidy -i --quiet true --tidy-mark true -wrap 0`

[logfile_dir]

Chemin. Le répertoire que ce chemin indique est celui dans lequel TYPO3 écrit les fichiers log selon la dénomination des serveurs Web, pour leur traitement par des programmes statistiques. Le serveur Web doit avoir le droit d'écrire dans le répertoire. Le nom du répertoire doit se terminer par une barre oblique. Vous trouverez plus d'informations à la section 4.12.2.

Exemple : `[logfile_dir] = /srv/www/logs/`

[logfile_write]

Il existe plusieurs méthodes pour écrire des fichiers log. Par défaut, TYPO3 utilise la commande UNIX `echo`. Si vous entrez `fputs`, TYPO3 utilise alors la fonction PHP du même nom qui fonctionne également en `safe_mode`.

Exemple : `[logfile_write] = fputs`

[publish_dir]

Chemin menant au répertoire dans lequel TYPO3 publie les pages HTML de manière

statique. Le serveur Web doit avoir le droit d'écrire dans le répertoire. Les pages peuvent alors être publiées à partir du panneau d'administration dans la zone **publish**.

Exemple : `[publish_dir] = /srv/www/htdocs/publish/`

`[addAllowedPaths]`

Chemins séparés par des virgules. Le stockage des ressources associées au TypoScript se fait dans les répertoires que vous indiquez ici. Les chemins sont relatifs au répertoire Web. Le répertoire par défaut commence par une barre oblique ; sans barre oblique, chaque répertoire qui commence avec la même expression est accepté.

Exemple : `[addAllowedPaths] = b2b/, /b2c/`

`[allowedTempPaths]`

Chemins séparés par des virgules. Il s'agit de chemins additionnels où vous placez des images temporaires pour l'utilisation de `imgResource` dans le TypoScript.

Exemple : `[allowedTempPaths] = b2btemp/`

`[debug]`

Booléen (0,1). Lorsque l'option est activée (1), les informations de débogage sont indiquées dans le frontend. Cela peut aussi être fait par TypoScript.

Exemple : `[debug] = 1`

`[simulateStaticDocuments]`

Booléen (0,1). Avec cette entrée, l'affichage d'adresses URL statiques simulées est activé par défaut, mais doit être configuré séparément par TypoScript.

Exemple : `[simulateStaticDocuments] = 1`

`[noPHPscriptInclude]`

Booléen (0,1). Lorsque cette option est activée, seuls les scripts PHP situés dans le répertoire `media/scripts/` sont appelés par TypoScript.

Exemple : `[noPHPscriptInclude] = 0`

`[compressionLevel]`

Cette valeur définit par la fonction `zlib` en PHP la compression des pages HTML dans le frontend. La valeur 1 correspond au taux de compression le plus faible et la valeur 9 au plus important. Plus le taux de compression est important, plus la bande passante est épargnée, mais plus la charge du serveur est grande. L'entrée **TRUE** permet au taux de compression d'être automatiquement adapté à la charge du système.

Exemple : `[compressionLevel] = 0`

`[compressionDebugInfo]`

Booléen (0,1). Les tailles des versions comprimées et non comprimées d'une page sont indiquées au bas de celle-ci lorsque vous activez cette option. Ceci devrait toutefois être uniquement utilisé pour des tests, étant donné que le contenu est comprimé deux fois afin d'afficher ces statistiques.

Exemple : `[compressionDebugInfo] = 0`

[pageNotFound_handling]

Chaîne de caractères. Grâce à cette option, vous spécifiez à TYPO3 comment réagir à des requêtes de pages lorsqu'elles ne sont pas disponibles. Le comportement par défaut est d'afficher la page « la plus proche ». En entrant `true` ou `1`, un message d'erreur s'affiche. Une alternative est de spécifier une page HTML.

Exemple : `[pageNotFound_handling] = http://www.brunching.com/gone.html`

[pageNotFound_handling_statheader]

Chaîne de caractères. Si l'option `[pageNotFound_handling]` est activée, la chaîne de caractères spécifiée est toujours envoyée comme en-tête.

Exemple : `[pageNotFound_handling_statheader] = Status: 404 Not Found`

[userFuncClassPrefix]

Ce préfixe est la première partie, soit de chaque fonction, soit du nom d'une classe appelée par TypoScript, par exemple dans la fonction `stdWrap`.

Exemple : `[userFuncClassPrefix] = user_`

[addRootLineFields]

Liste séparée par des virgules. Une liste de champs de la table `pages` est ajoutée à la requête de sélection.

Exemple : `[addRootLineFields] =`

[checkFeUserPid]

Booléen (0,1). Si l'option est activée, les formulaires d'identification dans le frontend doivent spécifier l'ID de la page (pid) sous lequel les utilisateurs frontend sont enregistrés. Si l'option est désactivée (0), la configuration `eval` de `uniqueInPid` dans le `$TCA` pour le champ `fe_users.username` devrait être changée en `unique`. L'entrée ressemble alors à ceci : `$TCA['fe_users']['columns']['username']['config']['eval'] = 'nospace,lower,required,unique'`; L'endroit de la sauvegarde n'est plus spécifié dans le TypoScript de la page sur laquelle le formulaire d'identification est situé ; tous les utilisateurs FE sont globalement valides.

Exemple : `[checkFeUserPid] = 1`

[defaultUserTSconfig]

Chaîne de caractères. Prédéfinissez les entrées `TSconfig` pour tous les utilisateurs frontend et les groupes.

Exemple : `[defaultUserTSconfig] =`

[defaultTypoScript_constants]

Chaîne de caractères. Cette option permet de prédéfinir les constantes TypoScript pour tout le système.

Exemple : `[defaultTypoScript_constants] =`

[defaultTypoScript_editorecfg]

Chaîne de caractères. Permet de définir la configuration `editorecfg` de TypoScript pour

tout le système. Cette option est utilisée par CSS Styler (clé d'extension : `ttemplate_cssanalyzer`).

Exemple : `[defaultTypoScript_editorcfg] =`

[dontSetCookie]

Booléen (0,1). Le système ne met aucun cookie dans le frontend lorsque cette option est activée, ce qui a pour effet d'empêcher les ouvertures de session pour l'identification.

Exemple : `[dontSetCookie] = 0`

[get_url_id_token]

Chaîne de caractères. Si l'option TypoScript `config.ftu` est activée, les utilisateurs, dans le frontend, peuvent ouvrir une session sans cookie. Dans ce cas, la session de l'utilisateur est gérée via un paramètre `get`. C'est le nom de ce paramètre que vous devez saisir ici. Ce type d'administration de session n'est en principe pas recommandé, car il mène plus facilement à des erreurs que la variante avec cookies.

Exemple : `[get_url_id_token] = SESSID`

[content_doktypes]

Chaîne de caractères. Définissez dans une liste séparée par des virgules les types de pages (valeur du champ `pages.doctype`) qui sont reconnues par le système comme pages ou dossiers système.

Exemple : `[content_doktypes] = 1,2,5,7`

[enable_mount_pids]

Booléen (0,1). Vous permet de désactiver globalement (0) la fonction de point de montage pour les pages.

Exemple : `[enable_mount_pids] = 1`

[pageOverlayFields]

Chaîne de caractères. Les champs spécifiés sont utilisés dans les requêtes de bases de données pour les sites Web multilingues. Cette option est pertinente pour les extensions qui ajoutent leurs propres champs à la table `pages`.

Exemple : `[pageOverlayFields] = title,subtitle,nav_title,media,keywords,description,abstr...`

[strictFormmail]

Booléen. La valeur 1 spécifie que la fonctionnalité « formmail » de TYPO3 envoie des emails seulement aux destinataires qui ont été encodés par le système. C'est une protection contre les personnes malveillantes qui pourraient détourner l'utilisation du formulaire mail.

Exemple : `[strictFormmail] = 1`

[secureFormmail]

Booléen. La valeur 1 spécifie que la fonctionnalité « formmail » de TYPO3 envoie des emails seulement aux destinataires qui sont définis dans l'enregistrement de l'élément

de contenu associé au formulaire. C'est une protection contre les personnes malveillantes qui pourraient détourner l'utilisation du formulaire mail.

Exemple : `[secureFormmail] = 1`

[lockIP]

Entier (0-4). Si cette valeur est supérieure à zéro, un contrôle est effectué sur le paramètre `REMOTE_ADDR_IP` des utilisateurs du frontend (`fe_users`) durant leur session. Cela améliore la sécurité, mais peut couper l'accès si l'utilisateur change d'adresse durant sa session (dans ce cas, choisissez une valeur plus basse : 2 ou 3). La valeur entière indique combien de parties de l'adresse IP doivent être incluses pour la vérification. Réduire la valeur à 1-3 signifie respectivement que la première, la deuxième ou la troisième partie de l'adresse IP est utilisée. 4 constitue l'entièreté de l'adresse IP et est la valeur recommandée. 0 (zéro) désactive la vérification.

Exemple : `[lockIP] = 2`

[loginSecurityLevel]

Chaîne de caractères. Voir la description de `TYPO3_CONF_VARS[BE][loginSecurityLevel]`. L'état par défaut du frontend est `normal`. D'autres niveaux plus élevés peuvent être configurés.

Exemple : `[loginSecurityLevel] = normal`

[lifetime]

Entier positif. Si la valeur est strictement supérieure à zéro, le cookie d'un utilisateur FE n'est pas un cookie de session (effacé lorsque la fenêtre du navigateur est fermée), mais plutôt un cookie avec une durée de vie indiquée par la valeur. Par exemple une valeur de `3600*24*7` aura pour résultat une identification automatique de l'utilisateur FE durant toute une semaine.

Exemple : `[lifetime] = 604800`

[maxSessionDataSize]

Entier. Spécifie la taille maximum (en octets) des données de sessions frontend qui sont sauvegardées dans la table `fe_session_data`. 0 signifie qu'il n'y a pas de limite. Cependant, cette valeur n'est pas conseillée, car on ne vérifie plus dès ce moment que les données de session sont enregistrées seulement après l'activation d'un cookie de confirmation.

Exemple : `[maxSessionDataSize] = 10000`

[lockHashKeyWords]

Liste de valeurs séparées par des virgules. La seule valeur est pour l'instant `useragent`. Cette valeur signifie que la session utilisateur de frontend dépend de la valeur du paramètre `HTTP_USER_AGENT`, diminuant ainsi le risque de détournement de session. Cependant, dans certains cas (tels que des solutions de paiement), vous devez désactiver cette option (p. ex. avec une chaîne de caractères vide) car le cookie de session est utilisé.

[hidePagesIfNotTranslatedByDefault]

2.5.6 Autres options

[MODS]: \$TYPO3_CONF_VARS["MODS"]

Contenait les options de configuration des modules, mais a été remplacée par le système d'extensions.

[USER]: \$TYPO3_CONF_VARS["USER"]

Contenait les options de configuration des paramètres de vos propres scripts mais a été remplacée par le système d'extensions.

[SC_OPTIONS]: \$TYPO3_CONF_VARS["SC_OPTIONS"]

Cette section est utilisée pour rendre disponibles vos propres options de configuration pour n'importe quel script dans TYPO3 (en général, des modules backend).

[EXTCONF]: \$TYPO3_CONF_VARS["EXTCONF"]

Vous pouvez rajouter ici vos options de configuration pour vos propres extensions. Pendant l'installation dans le gestionnaire d'extensions, celles-ci devront être affichées en utilisant le fichier `ext_conf_template.txt`.

Exemple : `$TYPO3_CONF_VARS['EXTCONF']['ma_cle_extension']['mon_option'] = 'ma_valeur';`

2.6 Séparation du serveur de production et du serveur en ligne

Lorsque vous avez besoin d'augmenter vos performances, et pour des raisons de sécurité, il peut être utile, et même nécessaire, de partager entre plusieurs serveurs la génération des pages et d'autres contenus ainsi que leur présentation.

Une possibilité est de publier le site Web en pages HTML statiques, sans avoir ni la base de données ni TYPO3 sur le système en ligne. Mais dans le scénario standard, le site Web lui-même contient ces éléments dynamiques générés à partir de la base de données, ce qui rend TYPO3 nécessaire sur le système en ligne. Le problème principal d'un tel scénario est la synchronisation du contenu de la base de données entre les différents systèmes. En effet, plusieurs serveurs pourraient être impliqués, par exemple pour répartir la charge.

Différents mécanismes de synchronisation sont disponibles, en fonction de la base de données utilisée. Consultez la documentation de votre vendeur de base de données.

La synchronisation directe ne vous permet pas de contrôler le processus de publication : ce qui se trouve sur le système de production est automatiquement placé en ligne. Une alternative est de réaliser manuellement la synchronisation, en fonction du SGBDR utilisé, pour permettre une étape supplémentaire dans la mise en ligne du contenu.

Pour l'installation de TYPO3 supportant le site en ligne et n'offrant pas de fonction d'édition, il est judicieux de désactiver les identifications des utilisateurs backend en donnant la valeur 1 au paramètre `$TYPO3_CONF_VARS['BE']['adminOnly']` de la section **All Configuration** de l'outil d'installation.

2.6.1 Pages statiques

Comme nous l'avons déjà mentionné brièvement à la section 2.5.5, une seconde possibilité, encore plus simple, est de publier des pages statiques. Une entrée dans l'outil d'installation, combinée à la configuration dans le panneau d'administration dans le frontend, permet de sauver toutes les pages Web dans un répertoire prédéfini du serveur.

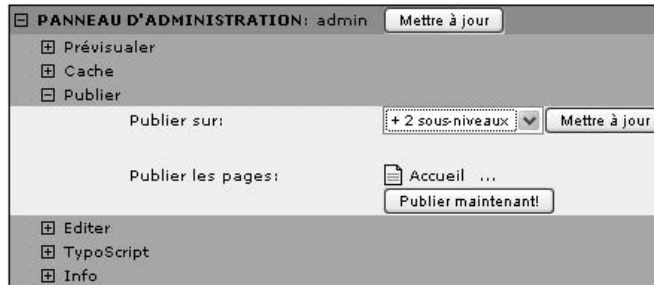


Figure 2.8:
Publication de pages
statiques sur
plusieurs niveaux
dans le panneau
d'administration du
frontend

Il existe d'autres logiciels qui permettent comme TYPO3 de publier des pages statiques. Par exemple, l'outil OpenSource HTTrack est disponible sous Linux et Windows et peut être téléchargé gratuitement à l'adresse : <http://www.httrack.com/>.

2.7 Sauvegardes

Il existe plusieurs méthodes pour créer des sauvegardes. La plupart des entreprises ont leurs propres stratégies de sauvegarde, et le moyen le plus sûr est d'inclure le répertoire adéquat du serveur Web dans ces routines de sauvegarde, sans oublier le répertoire où les fichiers de base de données sont enregistrés.

Il existe différentes extensions disponibles pour créer des sauvegardes à partir de TYPO3. Toutefois, même un administrateur relativement inexpérimenté peut facilement mettre en place une sauvegarde automatique au niveau du système d'exploitation, ce qui est une meilleure solution, puisque les sauvegardes ne sont plus lancées manuellement à chaque fois.

Avec une entrée dans le système crontab (UNIX), ce script est appelé régulièrement : dans l'exemple ci-dessous, il est lancé tous les jours à une heure du matin.

Dès lors, les fichiers de sauvegarde sont copiés automatiquement sur un système physiquement séparé. Un script tel que celui présenté ci-après réalise facilement et automatiquement cette opération grâce à `rsync`. Une clé SSH est nécessaire pour activer la transmission des données sous forme cryptée.

Voici un exemple de script¹² :

```
#!/bin/sh
# script rsync_backup.sh
# backup of webserver document root via rsync to backup server
# additional do a dump of typo3-db
```

¹²avec nos remerciements à Harald Oest de <http://www.ixsys.de>

```

# ip or fqhn of backup server
SERVER="my_backup_server"
# user account at backup server
USER="my_username"
# ssh-key (without passphrase!) used for login
SSHKEY="/root/.ssh/backup_server_key"
# destination dir at backup server
DSTDIR="/typo3_bkp"
# name of local typo3 database
DB="typo3_db"
# user account to access typo3 database
DB_USER="typo3_db_user"
# password to access typo3 database
DB_PASS="typo3_db_password"

# these directories will be rsynced with backup server
DIRS="/srv/www"

# do a mysql-dump and store result in source dir
/usr/bin/mysqldump --password=$DB_PASS -u $DB_USER $DB > \
    /srv/www/typo3db_bkp.sql

# rsync all requested dirs
for DIR in $DIRS; do
    logger "rsync backup $DIR to $SERVER"
    rsync --rsh="ssh -i $SSHKEY" -a $DIR $USER@$SERVER:$DSTDIR
done;

# get actual size of backup
ACT_SIZE = ssh -i $SSHKEY $USER@$SERVER "du -sh $DSTDIR"

logger "total backup size: $ACT_SIZE"

```

Vous générez une clé SSH sans demande de mot de passe comme suit :

```

~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/backu
p_server_key
Enter passphrase (empty for no passphrase): [Enter]
Enter same passphrase again: [Enter]
Your identification has been saved in /root/.ssh/backup_server_key.
Your public key has been saved in /root/.ssh/backup_server_key.pub.
The key fingerprint is:
8b:1f:f9:c9:54:65:bc:f5:d6:ce:79:0d:e4:1d:56:2f root@local_box

```

La paire de clés a maintenant été créée et la clé privée est déjà au bon endroit (/root/.ssh/backup_server_key). Il reste à envoyer la clé publique au serveur de sauvegarde :

```

~# scp /root/.ssh/backup_server_key.pub \
> user@my_backup_server:/home/.ssh

```

À présent, connectez-vous à my_backup_server et activez la clé SSH :

```
root@my_backup_server:~ # cat /home/.ssh/backup_server_key.pub >> \
> /home/.ssh/authorized_keys
```

Pour l'effacer :

```
root@my_backup_server:~ # rm /home/.ssh/backup_server_key.pub
```

La même chose sur l'ordinateur local :

```
~#rm /root/.ssh/backup_server_key.pub
```

Vous pouvez maintenant vous connecter au serveur de sauvegarde sans mot de passe :

```
~#ssh -i /root/.ssh/backup_server_key user@my_backup_server
```

Voici un exemple d'une entrée crontab qui appelle la sauvegarde rsync à une heure du matin :

```
# call backup script every night at 01.00
0 1 * * * root test -x /root/bin/rsync_backup.sh && \
    /root/bin/rsync_backup.sh
```

2.8 Mises à jour

Les mises à jour de TYPO3 sont particulièrement agréables pour l'administrateur car elles durent rarement plus de quelques minutes, et ceci grâce aux liens symboliques déjà souvent mentionnés. Ceux-ci pointent vers un répertoire unique qui contient le code source de TYPO3 ; tous les autres fichiers sont soit indépendants de la version et restent où ils sont, soit sont également des liens symboliques.

La première étape de toute mise à jour est de sauvegarder l'ensemble de l'installation, ou du moins la base de données. Avec la commande :

```
~#/srv/wwwmysqldump -u user -p nom_base_données backup.tgz
```

vous créez un fichier de sauvegarde après avoir entré le mot de passe de l'utilisateur de la base de données de MySQL. En cas d'urgence, vous pouvez désarchiver cette sauvegarde grâce à la commande :

```
~#/srv/wwwtar xzf backup.tgz
```

et ensuite, par la commande :

```
~#/srv/wwwmysql -u user -p datenbankname < backup.sql
```

restaurer la sauvegarde dans la base de données.

Retour à la mise à jour : pour utiliser la nouvelle version plutôt que l'ancienne, supprimez l'ancien lien symbolique. Par exemple, dès que vous entrez :

```
~#/srv/wwwrm typo3_src
```


votre site n'est plus en ligne !

Avec :

```
~# /srv/wwwln -s ../typo3-src-4.0 typo3_src
```

un nouveau lien symbolique pointe vers la nouvelle source que vous venez de placer sur le serveur (cf. section 2.3.1).

Sous Windows et généralement avec des distributions zip, tous les répertoires TYPO3 (typo3, t3lib, tslib) doivent être remplacés manuellement, si vous n'utilisez pas Junction.

Ainsi, la mise à jour est déjà terminée au niveau des fichiers si le lien symbolique pointant vers la version TYPO3 est recréé.

Rappelez-vous bien qu'après un changement de version de TYPO3, vous devriez d'abord appeler le Database Analyzer de l'outil d'installation. Les changements nécessaires à la base de données sont identifiés et affichés à l'aide de la fonction **COMPARE**. Ces changements concernent l'ajout de nouveaux champs dans la base de données ou des changements de définition de champs existants. Sélectionnez **Write to database** et exécutez la mise à jour. La fonction **COMPARE** enlève également les tables de bases de données associées à des extensions qui ont été désinstallées. Il va de soi que vous devez savoir ce que vous faites lorsque vous exécutez cette fonction, et bien sûr : sauvegardez, sauvegardez, sauvegardez ! Les tables et les champs à enlever sont d'abord renommés en ajoutant à leur nom le préfixe **zzz_**. Ils sont réactivés en enlevant ce préfixe, par exemple avec **phpMyAdmin**.

Il est conseillé, pour des raisons évidentes, de ne pas exécuter la mise à jour sur des sites très visités. Dans ce cas-là, il est préférable de faire une copie de la base de données et des fichiers, de définir l'accès à la nouvelle base de données et d'exécuter des scénarios de reprise de données. Cela vous prépare par ailleurs à migrer en quelques secondes vers un nouveau serveur ou à changer de disque dur.

2.9 En cas de problème

La communauté TYPO3 vous propose plusieurs types d'aide en cas de problème d'installation. Ces problèmes sont très diversifiés étant donné les différentes combinaisons possibles de serveur Web, de base de données et de distribution PHP utilisés. Deux « mailing lists » existent pour les questions d'installation, l'une pour le système d'exploitation Linux et l'autre pour Windows, cette dernière étant aussi utilisée via le protocole nntp (*newsgroups*). En outre, il existe une archive en ligne que vous devriez certainement consulter afin de vous assurer que votre question n'a pas déjà été posée. Souvenez-vous que les membres de la communauté d'utilisateurs de TYPO3 qui vous aident sont des volontaires : en leur demandant de l'aide, ne les distrayez pas avec des questions inutiles. Si vous découvrez un bogue qui n'a pas encore été décrit, enregistrez-vous à l'adresse <http://bugs.typo3.org>.

TYP03 pour les rédacteurs

- Elle doit être cohérente et garder une structure uniforme ;
- Elle doit offrir la possibilité de la personnaliser suivant le rôle réservé aux différents utilisateurs ou groupes d'utilisateurs ;
- Chaque fonction de l'application doit proposer à l'utilisateur des outils d'aide contextuels, un guide d'utilisation et des exemples pour débutants ;
- L'interface doit offrir la possibilité d'annuler des manipulations individuelles opérées au cours d'une session, simplement en poussant sur un bouton (fonction « Annuler »). Un historique des modifications permet de recréer les étapes de travail précédemment enregistrées.
- De plus, les assistants (*wizards*) sont importants afin de guider les utilisateurs lors d'opérations spécifiques.

En règle générale, un manque ou un excès de fonctionnalités nuit au bon déroulement du travail. Il est donc nécessaire d'avoir la possibilité d'étendre ou de réduire les fonctionnalités : le système doit fournir exactement aux utilisateurs les fonctions nécessaires à leur travail. De plus, les connaissances techniques requises pour utiliser le CMS doivent être minimales, afin de réduire le plus possible l'effort d'installation.

TYP03 offre tous ces avantages, et plus encore. Au cours de ses huit années de développement, la convivialité de l'interface utilisateur a été optimisée grâce aux expériences des particuliers qui l'utilisent au quotidien (environ 122 000 installations sont actuellement opérationnelles).

3.1 Le rôle du rédacteur

La tâche du rédacteur est de créer de nouveaux contenus, de les adapter au média auquel ils sont destinés, et de les intégrer dans l'application correspondante, un site Web par exemple. L'administrateur place alors l'utilisateur, par le biais des droits d'accès, dans une sorte de « bac à sable » à partir duquel il a un accès (limité) au système et à ses fonctionnalités. Dès lors, l'utilisateur accède à son espace de travail (backend) qui correspond à ses tâches et aux ressources qui lui sont allouées (voir chapitre 4.1). Alors qu'un rédacteur de presse, par exemple, introduit³ de nouveaux articles dans le système, un rédacteur en chef aura une vue plus étendue, de par ses droits d'accès associés à son groupe d'utilisateur.

Référence 509060

En fonction de la durée du cycle de vie du contenu et de la complexité des procédures de travail, le rédacteur se contente de transformer certaines parties du contenu sur l'itinéraire vers sa publication. Le rédacteur travaille au sein de ce qu'on appelle un *workflow*, avec d'autres rédacteurs.

Référence 509060

Nous allons vous présenter ci-après l'interface utilisateur dans laquelle le contenu est rédigé, ainsi qu'un module de tâches — une sorte de centre de communication pour les membres d'un projet. Nous poursuivrons en expliquant la façon de créer une page et d'y ajouter un contenu. Nous ne nous attarderons pas sur chaque champ (des rubriques d'aide contextuelles ainsi que le très complet « Manuel du rédacteur » sont prévus à cet effet, cf. référence), mais plutôt sur l'aspect général des pages et des types de contenu, ainsi que sur leur utilisation pratique. Les

³ « *works in* » dans le jargon anglais des habitués du CMS

aspects spécifiques tels que la manipulation de *Rich Text Editor*, un éditeur de texte, seront traités séparément. Finalement, nous présenterons les procédures et les outils qui simplifient et rendent plus efficace le travail quotidien des éditeurs sur TYPO3.

3.2 Accéder au système

TYPO3 est un système de gestion de contenu basé sur une interface Web. Il est soit installé sur le serveur intranet dans une entreprise, soit sur un serveur Web ou extranet accessible publiquement. Pour accéder au système et créer ou mettre à jour le contenu d'une page, les rédacteurs n'ont besoin que d'un navigateur et de l'adresse Internet correcte, peu important l'endroit et l'heure.

TYPO3 fait la distinction entre deux zones différentes : le *backend* ou zone d'administration pour la gestion des pages et des données, et le *frontend*, le site Web tel que le voient les visiteurs. Le rédacteur peut utiliser ces deux modes de TYPO3 pour modifier le contenu. Alors que le backend offre un environnement de travail complet, comportant de nombreuses fonctionnalités, le frontend peut être utilisé de façon très simple et intuitive. Les outils d'édition de base sont fort semblables dans les deux zones ; les autres éléments et la manière de travailler décrits ici se retrouvent partiellement dans le frontend.

3.2.1 Configuration du navigateur

TYPO3 fonctionne avec différents navigateurs (Netscape Navigator 6.x, 7.x, Mozilla Firefox, Internet Explorer 5.x, 6.x, etc.) qui varient légèrement dans leur affichage et leurs fonctionnalités. Ces différences ne sont pas significatives, à ceci près que Rich Text Editor n'est disponible qu'avec Internet Explorer, parce qu'il nécessite ActiveX de Microsoft. Il existe plusieurs alternatives pour Mozilla via l'installation d'extensions dans le système.

Référence 352869

Le navigateur doit accepter les cookies, et le JavaScript doit être activé. La configuration du cache du navigateur doit être ajustée de sorte que la toute dernière version d'une page soit servie lorsqu'on en fait la requête.

3.2.2 Identification

Pour vous identifier afin d'accéder au backend, il vous suffit d'entrer l'URL de votre site dans votre navigateur suivi de */typo3* (ex. : <http://www.votredomaine.com/typo3>). La fenêtre d'identification (voir image) apparaît. Entrez votre nom d'utilisateur et votre mot de passe. Les mots de passe pour l'accès au backend sont transmis sous forme encryptée par TYPO3 et sauvegardés en toute sécurité dans le système de base de données.



Figure 3.1:
Identification avec
nom d'utilisateur et
mot de passe

3.3 Interface utilisateur et modules

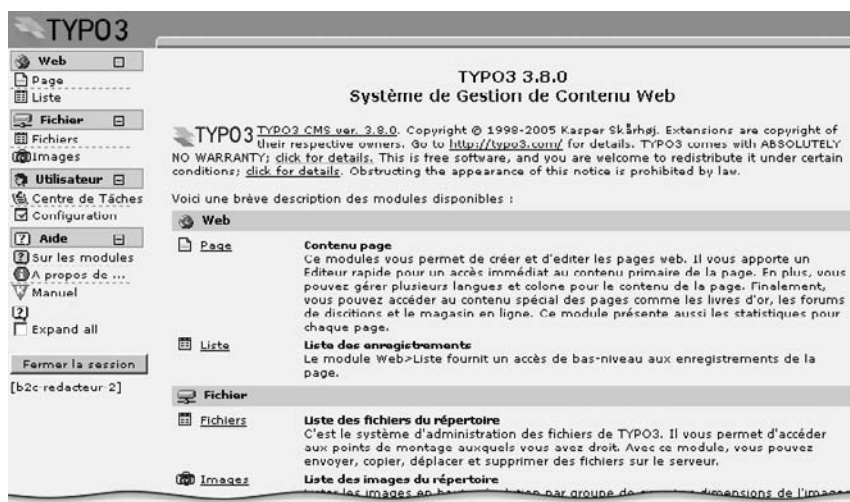
Référence 502403

Le backend désigne l'interface utilisateur de TYPO3 où travaille le rédacteur. L'administration des pages et de leur contenu via l'interface frontend sera reprise plus bas (cf. section 3.8). Une fois que le rédacteur s'est identifié, il accède à son espace de travail. N'oubliez pas : l'apparence de l'interface utilisateur peut varier considérablement d'un rédacteur à un autre. Tout dépend de la configuration définie par l'administrateur.

Juste après l'identification, vous vous trouvez ensuite soit sur une page d'aide, présentant une courte description de chaque module, soit dans le module **Utilisateur** → **Centre de tâches**.

En cliquant sur une des options de navigation du menu **Web** dans la partie gauche de la page, la structure du backend s'affiche ainsi :

Figure 3.2:
Module Aide → sur
les modules après
identification



L'interface utilisateur de TYPO3 est généralement divisée en trois colonnes : le *menu de modules* à gauche, l' *aire de navigation* avec l'arborescence au milieu, et la *vue détaillée* à droite.

3.3.1 Zones de l'interface utilisateur

Si plusieurs rédacteurs différents travaillent sur le même système, le module **Utilisateur** → **Centre de tâches** (le centre de communication et d'administration de TYPO3) présente normalement une configuration par défaut lorsqu'il est utilisé pour la première fois. On y retrouve les trois zones caractéristiques de l'interface utilisateur.



Figure 3.3:
Les trois zones du bureau dans le module Tâches : menu de modules [1] aire de navigation [2] vue détaillée [3]

Menu de modules

Le menu de modules constitue le menu principal du backend de TYPO3. Il reprend les modules auxquels un rédacteur particulier a accès.

Les modules principaux (**Web**, **Fichier**, **Doc**, ...) ne sont qu'un groupement de sous-modules (ex. : **Page**, **Voir**, **Liste**, etc.) qui, eux, pointent vers les véritables outils d'édition du backend.

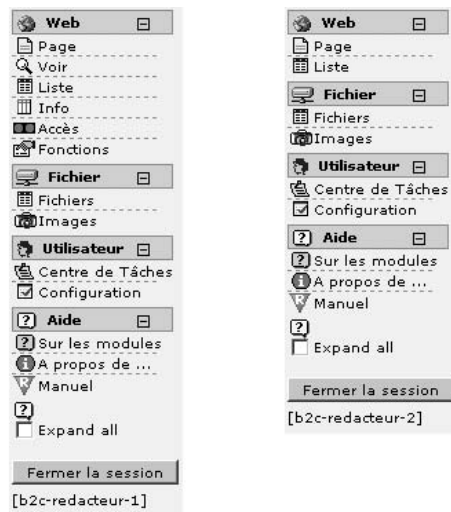


Figure 3.4:
Configurations backend : deux utilisateurs ayant des droits d'accès différents

Aire de navigation

Après avoir sélectionné un sous-module, la navigation se fait dans l'*aire de navigation*. Dans le module principal **Web**, la colonne centrale affiche l'*arborescence*, où sont reprises toutes les pages auxquelles le rédacteur a accès (fg. 3.5[1]).

On accède à l'arborescence des répertoires, donnant l'accès au gestionnaire de fichiers interne à TYPO3, via le module principal **Fichier** (fg. 3.5[2]).

À l'instar de l'explorateur Windows et d'autres gestionnaires de fichiers, les niveaux de la structure des répertoires peuvent être étendus à l'aide des icônes « + » et « - ». Si vous cliquez sur le titre, une vue détaillée du répertoire sélectionné s'affichera à droite.

Si vous cliquez sur une icône représentant un dossier, un menu contextuel s'affiche afin de permettre un accès rapide aux fonctions principales de travail. Dans les navigateurs Windows, ce menu s'ouvre directement dans l'arborescence. Pour les autres navigateurs, les fonctions apparaissent dans le cadre supérieur du backend.

Gardez à l'esprit les règles suivantes :

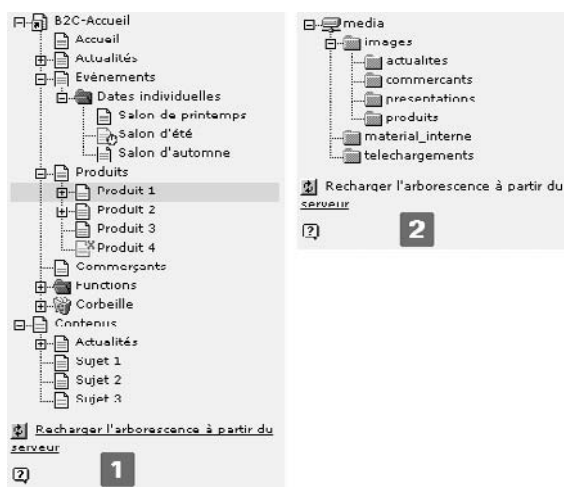
Cliquez sur l'*icône* =

Appel des fonctions du fichier (copier des éléments, couper etc.)

Cliquez sur le *titre* =

Visualisez/modifiez le contenu

Figure 3.5:
Arborescence des
pages [1] et
arborescence des
répertoires du
gestionnaire de
fichiers [2]



Vue détaillée

Le véritable espace de travail se trouve dans la *vue détaillée* (troisième colonne). Une multitude d'informations peuvent s'y afficher en fonction du module sélectionné. Dans les modules principaux **Web** et **Fichier**, un menu contextuel est aussi disponible, afin de simplifier votre travail (rappelez-vous : cliquez sur l'icône).

3.3.2 Modules principaux et sous-modules – un aperçu

Web

Le module **Web** est celui dont le rédacteur se sert le plus. Les pages et leur contenu y sont enregistrés, gérés et modifiés. Les sous-modules s'y rapportant vous offrent souvent plusieurs possibilités pour arriver à des résultats identiques.

Référence 762976

Page

Le module **Page** permet de créer et de modifier de nouvelles pages et leur contenu.

Les options disponibles sont :

Édition rapide

Le contenu est affiché directement en mode d'édition. Le menu déroulant ou la barre de bas de page vous donnent la possibilité de passer d'un élément en cours de modification à un autre.

Colonnes

Affiche le contenu de toutes les colonnes d'une page.

Langues

Représente le contenu d'une page classé selon la langue dans laquelle celle-ci est définie. La gestion des traductions d'une page ne pose pas de réel problème car TYPO3 permet l'intégration d'un site multilingue dans une seule et même arborescence.

Informations sur la page

Fournit une vue d'ensemble des informations de base de la page ; vous pouvez par exemple voir quelle est la configuration du cache pour la page, si elle doit être publiée pendant une période déterminée, si son accès n'est réservé qu'à certains utilisateurs frontend, ou quelle méta-information à l'intention des moteurs de recherche se trouve sur la page.

Voir

Le module **Voir** affiche une page exactement comme un visiteur la verra sur le site. La page affichée comporte aussi des petites icônes « crayon » qui permettent d'éditer la page directement. Ce module vous permet de visualiser des pages n'ayant pas encore été mises en ligne.

Liste

L' « affichage liste » donne un accès direct aux pages et à leur contenu. Les utilisateurs expérimentés préfèrent généralement travailler dans ce module. En activant la **Vue étendue** et le **presse-papiers** à l'aide des cases à cocher situées au bas de la vue détaillée, combinés à une variété de fonctions, il est possible d'éditer les enregistrements simultanément, de façon à atteindre des objectifs précis. Vous trouverez plus d'informations sur cette fonction à la section 3.10. La **Vue de localisation** donne une vue d'ensemble des différentes traductions des éléments de contenu.

Info

Dans le module **Info** se trouve un résumé des informations importantes à propos de la page sélectionnée. En fonction de ses droits d'accès, le rédacteur y trouve différentes vues de la base de données :

Arborescence (vue d'ensemble)

Contient les paramètres de base (ex. : publication limitée dans le temps ou droits d'accès pour des groupes frontend), le cache et l'âge, ainsi qu'un aperçu des types d'enregistrements contenus dans les pages pour lesquelles le rédacteur a les droits d'édition. Si vous utilisez cette vue sur plusieurs niveaux de l'arborescence, les valeurs d'un champ sur un ensemble de pages peuvent être éditées en un seul clic, en utilisant l'icône « crayon ».

Localisation (vue d'ensemble)

La visualisation des traductions vous permet d'accéder à plusieurs niveaux de page pour vérifier l'état des traductions. En même temps, vous pouvez créer des en-têtes pour de nouvelles pages de traduction, ou encore éditer ou créer de nouvelles traductions de contenus.

Fichier journal

Ce module garde une trace de l'ensemble des modifications apportées par tous les rédacteurs ; la vue peut être ajustée dans la barre de navigation en fonction du nombre de niveaux à afficher et de la période de temps sur laquelle les modifications s'échelonnent. Dans la colonne **Détails**, l'historique des modifications individuelles peut être affiché et ouvert en formulaire d'édition à l'aide du lien – → **His**.

Configuration TS de la page

La **Configuration TS de la page** n'a pas beaucoup d'importance pour le rédacteur, et devrait normalement être désactivée par l'administrateur ; elle montre les détails TypoScript spécifiques aux pages et aux utilisateurs, qui sont importants pour les administrateurs et les développeurs (la règle suivante s'applique ici : ce qui a été déterminé pour une page vaut pour toutes les pages qui lui sont subordonnées).

Statistiques d'affichage

Les **Statistiques d'affichage** reprennent les données relatives aux visites sur votre site. Si l'extension **Simple hit statistics** est installée sur votre système, l'administrateur peut rapidement trouver un bon résumé du taux de fréquentation de chaque page du site.

Accès

Si le rédacteur a accès à ce module, il peut afficher les permissions de chaque page via le mode **Utilisateur**. Si le rédacteur est le « propriétaire » d'une page, c'est-à-dire qu'il l'a créée, il peut (et dans ce cas uniquement) en transférer les droits ou les retirer à un groupe dont il est membre, ou encore transférer sa propriété de la page à un autre membre du groupe. Le mode **Permissions** dresse la liste des propriétaires, du groupe dont il fait partie, et des permissions pour chacun. Le rédacteur ne peut administrer ici que les droits des pages dont il est propriétaire.

Fonctions

Le module **Fonctions** contient des outils simplifiant la création et l'administration de pages et de contenu. Il est bien sûr nécessaire que les extensions correspondantes aient préalablement été installées et mises à la disposition du rédacteur. Les divers **Assistants** de ce module permettent de manipuler l'arborescence des pages pour la création ou le tri de pages. Le mode **Importer** permet d'importer des parties entières d'arborescence TYPO3. L'outil **Text tools** effectue des recherches et/ou des remplacements des extraits de textes dans les éléments de contenu.

Nous aborderons l'utilisation de ce dernier outil à l'aide d'un exemple pratique à la section 3.10.

Fichier

TYPO3 gère toutes les ressources (telles que les gabarits HTML, les images ou les documents) dans le répertoire `fileadmin/` de son gestionnaire de fichiers. En fonction de ses tâches, le rédacteur reçoit des droits d'accès à différents points de montage du gestionnaire de fichiers. De plus, l'administrateur peut déterminer précisément le type d'accès : par exemple, si le rédacteur a le droit d'envoyer vers le serveur, de déplacer, d'effacer, de renommer, d'éditer ou de créer de nouveaux fichiers, voire de supprimer un répertoire et ses sous-répertoires.

Fichiers

Le sous-module **Fichiers** permet au rédacteur d'accéder aux ressources susmentionnées. Le contenu du dossier sélectionné s'affiche dans la vue détaillée. Vous pouvez aussi naviguer via les icônes dossier de l'arborescence. Pour rendre les fichiers accessibles, il suffit, à l'aide du menu contextuel, de les envoyer directement du PC du rédacteur ou du réseau local vers son répertoire respectif dans TYPO3. Encore une fois la règle suivante s'applique :

Cliquez sur l'*icône* =

Appelle les fonctions du fichier (Copier, couper etc.)

Cliquez sur le *titre* =

Visualiser/éditer le contenu

Les cases à cocher situées en bas de la vue détaillée activent l'affichage de vignettes pour les fichiers image et le presse-papiers reprend le contenu de la mémoire tampon pour copier ou déplacer des fichiers.

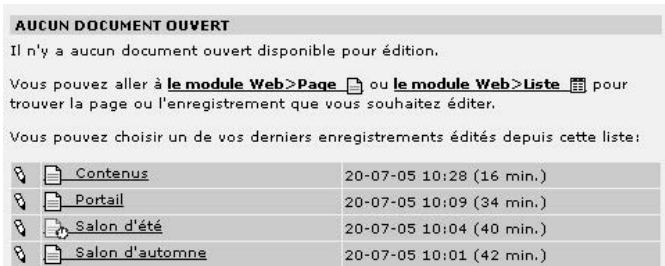
Images

Le sous-module **Images** reproduit les fichiers en mode prévisualisation, accompagnés de détails sur la taille de l'image et de ses dimensions en pixels.

Doc

Le module **Doc** permet d'accéder à des pages ou des éléments de contenu que vous avez ouverts afin de les éditer. Si vous avez une série d'enregistrements ouverts, ce module vous permet de passer rapidement d'un formulaire d'édition à un module. Par conséquent vous pouvez, par exemple, passer efficacement du gestionnaire de fichiers au formulaire d'édition d'un élément de contenu. Vous pouvez naviguer d'un enregistrement à un autre via le menu déroulant. Si aucun document n'est ouvert, une liste des derniers enregistrements modifiés s'affiche.

Figure 3.6:
Affichage des derniers
enregistrements
modifiés dans le
module Doc



Utilisateur

Le module **Utilisateur** montre les tâches et la configuration de l'interface propres à un utilisateur. Il offre aussi la possibilité de communiquer entre utilisateurs.

Centre de tâches

Sous l'appellation « centre de tâches » se cache le centre de communication de TYPO3. Plusieurs sous-modules permettent, par exemple, de sauvegarder des notes personnelles ou encore de communiquer entre membres d'un même projet à l'intérieur de workflows et du système de messagerie. Nous décrivons le module plus en détail à la section 3.4

Configuration

Référence 582216

Chaque utilisateur a la possibilité, dans une certaine mesure, d'adapter son environnement de travail de TYPO3 à ses besoins, à commencer par la sélection de la langue dans laquelle le backend est présenté. De plus, vous pouvez ajuster l'affichage du backend en fonction de vos préférences ou des performances de votre écran. De cette façon, le menu de modules peut être affiché sous forme de boîte de sélection ou sous forme d'icônes dans le cadre supérieur du backend, afin de gagner plus d'espace. Vous pouvez décider d'afficher les rubriques d'aide pour les champs de saisie individuels, ou si vous le souhaitez, vous pouvez décider que les images listées s'affichent automatiquement sous forme de vignettes.

Le menu contextuel ou le Rich Text Editor (pour les éléments de contenu **Texte** ou **Texte & image**) peuvent être désactivés, les copies récursives limitées à un certain nombre de niveaux, ou l'effacement récursif totalement interdit ou désactivé. Il est de toute façon recom-

mandé d'activer le chargement de fichiers vers le serveur via le module **Doc** et le navigateur d'éléments afin de pouvoir travailler efficacement : de cette façon, vous envoyez les fichiers à partir des modules où vous travaillez, sans passer par le module **Fichier**. L'administrateur peut, de manière centralisée, configurer l'interface backend pour chaque utilisateur et chaque groupe. La validité de certains paramètres peut être limitée dans le temps (cf. chapitre 4.8.2).

Si l'administrateur ne l'a pas déjà fait, vous devez saisir votre nom et adresse email dans la section « Données personnelles ». Certaines extensions, telles que le module **News**, utilisent ces informations afin de déterminer automatiquement l'auteur des nouvelles entrées. Cochez la case en bas du formulaire afin d'être informé par email chaque fois que quelqu'un s'identifie sous votre nom d'utilisateur.

Aide

Le module **Aide** peut comprendre plusieurs sous-modules. En règle générale, les sous-modules **Sur les modules**, **À propos de TYPO3** et **Manuel** sont activés ; le premier présente la fonction principale de chacun des modules, le deuxième contient des informations à propos de la version utilisée et des droits d'auteur, et indique les termes de la licence de TYPO3.

Référence 647456

Afin que la description de toutes les sections relevant du rédacteur ainsi que de leurs options d'entrée soit toujours disponible, le « Manuel du rédacteur » peut être intégré dans le backend à titre d'extension.

3.4 Le module utilisateur → centre de tâches comme centre de communication

Ce module de TYPO3 propose plusieurs outils, facilitant l'édition de contenu dans le cadre plus large d'un workflow, et permettant d'accéder rapidement à certaines actions et à certains éléments de contenu à partir de l'arborescence des pages. Il offre aussi des fonctions importantes pour simplifier et concentrer les tâches dans une partie du système.

Si le module est appelé via **Utilisateur → Centre de tâches**, l'aire de navigation affiche les fonctions de sélection. Nous les reprenons ici en détail dans l'ordre où elles apparaissent.

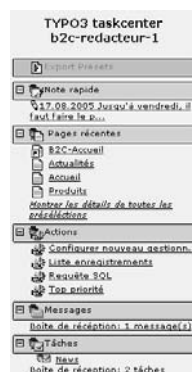


Figure 3.7:
Aire de navigation du
module Utilisateur →
Centre de tâches
(exemple)

Note rapide

Référence 740912 Le champ de saisie offre à l'utilisateur backend un espace pour ses commentaires, ses notes ou d'autres textes. Leur présence sera rappelée à l'utilisateur lors de l'ouverture de sa prochaine session. Ces entrées ne sont pas visibles pour les autres utilisateurs (à l'exception de l'administrateur).

Pages récentes

La fonction **Pages récentes** dresse, dans l'aire de navigation, une liste de liens vers les dernières pages éditées par l'utilisateur. Si vous cliquez directement sur **Montrer les détails de toutes les présélections**, une liste des enregistrements fréquemment édités s'affiche, dans l'ordre chronologique de leur modification.

Actions

Référence 748421 L'administrateur a la possibilité d'assigner des actions prédéfinies à certains utilisateurs ou à des groupes d'utilisateurs à l'aide des « actions ». Les types d'actions repris ci-dessous font référence à leur fonction, et non au nom sous lequel l'action est affichée dans le module **Utilisateur** → **Centre de tâches**.

Créer un utilisateur backend

Un rédacteur peut créer ici un nouveau compte utilisateur backend avec seulement quelques informations et sans les droits d'administrateur. L'administrateur peut assigner lui-même un nom aux commandes (ex. : « Créer un nouveau responsable produits » en utilisant la commande de type « Créer un utilisateur backend »).

SQL Query

Cette fonction est utile si vous désirez obtenir dans un format prédéterminé des données en provenance directe de la base de données, par exemple, les enregistrements de produits ou de transactions, puis les exporter en format CSV ou XML afin de les utiliser dans d'autres applications.

Record list

Elle montre les enregistrements associés à différents endroits de l'arborescence des pages, afin d'y accéder directement dans la vue détaillée du module **Web** → **Liste**.

Edit records

Cette fonction crée une liste d'éléments (page, contenu, autre enregistrement) depuis n'importe quelle position dans l'arborescence, et les rend disponibles pour un traitement rapide.

La création d'actions est vue plus en détail à la section 4.10. Lorsque vous entreprenez une action, la vue détaillée vous montre le type d'action, la description de celle-ci, ainsi que l'enregistrement qui sera créé ou modifié.

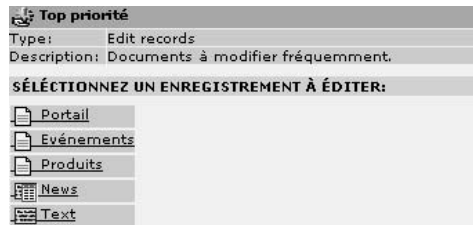


Figure 3.8:
Vue détaillée d'une
action d'édition de
page

Messages

Dans la section **Messages**, le rédacteur retrouve un système simplifié de messagerie pour la communication interne au projet. Le rédacteur peut l'utiliser pour envoyer (ou recevoir) des messages internes à des membres individuels ou à des groupes. Les nouveaux messages sont affichés dans le module **Utilisateur → Centre de tâches** et enregistrés dans la boîte de réception [1]. Les messages qui y sont enregistrés peuvent être déplacés vers une archive ou effacés [2]. Si un message doit être lu par un utilisateur avant son prochain accès au backend, il est possible de le lui envoyer directement sur son adresse email. Cela implique bien sûr que le serveur Web ait aussi un serveur email. On peut répondre aux messages reçus [3] en reprenant l'original [4].



Figure 3.9:
Vous avez un
message : boîte de
réception [1],
déplacer ou effacer
les emails [2],
contenu du message
[3], répondre au
message [4]

Tâches

La section **Tâches** contient les workflows que l'administrateur définit pour les rédacteurs.⁴ Dans la barre de navigation sont affichées les tâches « à faire ». En fonction des droits accordés à l'intérieur d'un workflow, un rédacteur peut être autorisé à initier de nouvelles tâches. Un auteur peut uniquement effectuer un changement de statut dans le cadre d'une étape bien définie du processus, ou autoriser un enregistrement pour sa publication.

La vue détaillée (troisième colonne) reprend vos **éléments à faire**, vos **éléments à faire pour d'autres utilisateurs** [2] —pour autant que le rédacteur ait les permissions requises— et l'affichage des détails pour l'enregistrement sélectionné [3]. Le **log de status** indique toutes les étapes par lesquelles l'enregistrement est passé jusqu'à son état actuel, avec la possibilité d'éditer l'enregistrement. Si le rédacteur a modifié son enregistrement, ou si certains obstacles surgissent, il y ajoute alors un nouveau statut accompagné d'un commentaire. L'enregistrement passe alors une étape supplémentaire dans le workflow [5]. Seuls les gestionnaires de tâches sont habilités à initier un nouveau workflow [6].

Figure 3.10:
Édition d'un
enregistrement à
partir du workflow
via le module Centre
de tâches

VOS ÉLÉMENTS 'A FAIRE'

Titre:	Workflow:	Objectif:	Effectué:
Actualités	Actualités	03-08-05 12:29 (5 j.)	

Vos éléments 'A Faire' pour d'autres utilisateurs:

Titre:	Workflow:	Objectif:	Effectué:
Démarrer la maquette	Actualités	28-09-05 15:39 (61 j.)	

DÉTAILS DES TÂCHES

Actualités

Créer par: portail-redacteur (j., 20-07-05 12:31 (-9 j.)
 Objectif: 03-08-05 12:29 (5 j.)
 Description: Bonjour redacteur 1

Pouvez-vous insérer une nouvelle page pour le salon à Londres. Le matériel graphique est déjà disponible dans le répertoire convenu.

Workflow: Actualités
 Description du workflow: Workflow pour les actualités

Log de status (Instance #8):

Information de status:	Commentaires et détails:
Ajouter un commentaire b2c-redacteur-1 (b2c-redacteur-1) 20-07-05 12:48 (9 j.)	Bonjour, la m'y mets immédiatement

L'instance a pour cible: **b2c-redacteur-1 (b2c-redacteur-1)**

Enregistrement en relation avec cette instance:

Ajouter un status:
 Ajouter un commentaire:
 Sélectionner l'utilisateur/groupe cible:
 Status (note):

CRÉER UNE NOUVELLE TÂCHE 'A FAIRE'

Workflow:

⁴La création de workflows est décrite à la section 4.9.

3.5 Les pages, réceptacles de contenu

3.5.1 Structure d'un site, arborescence et éléments de contenu

Avant de présenter les tâches principales du rédacteur (créer des pages et du contenu), nous développons d'abord le concept de base de TYPO3. Nous passons par cette étape, non pas pour aborder les aspects techniques de TYPO3, mais principalement pour a) bien manipuler la notion d'arborescence lors de la création d'un site, cette dernière se matérialisant sous la forme de menus dans le frontend, et b) comprendre la relation entre les éléments de contenu et la page qui les contient.

Pour illustrer notre propos, nous avons choisi comme exemple une petite application du site B2C classique. Celle-ci contient déjà plusieurs pages de différents types (voir fg. 3.11). Nous modifierons graduellement cette application au cours des explications. Nous vous encourageons à l'utiliser afin de vous exercer. À la section 5.9, une variante de l'application du site B2C nous servira de base pour expliquer la configuration et les développements plus avancés du frontend. L'arborescence du site est constituée de pages individuelles, telles que « Accueil », « Produit1 », « Contacts », et des sections principales telles que « Actualités », « Événements », et « Produits ».

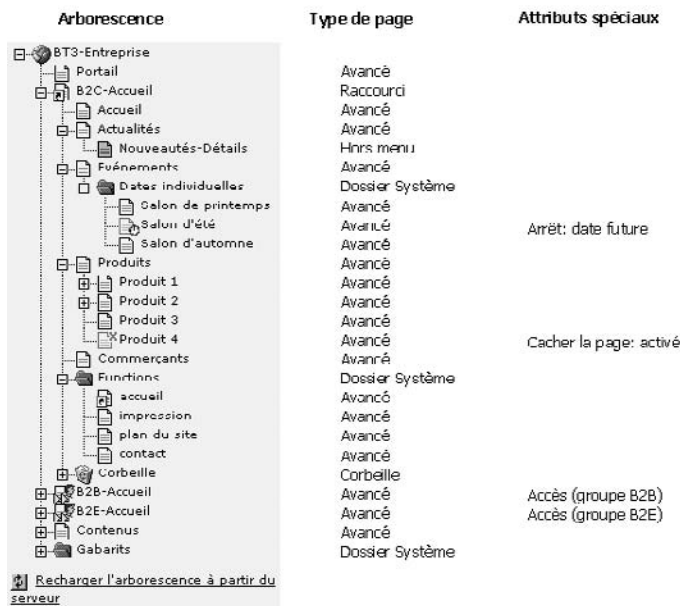


Figure 3.11:
Arborescence du site
exemple

La page en tant que telle ne contient pas l'information ! Elle sert simplement de réceptacle aux éléments de contenu qui lui sont assignés. Le rédacteur travaillant dans TYPO3 doit rarement s'inquiéter de questions techniques ; tout ce qui lui importe est de connaître la position, l'ordre et le type dans lesquels il doit créer de nouvelles pages et sous-pages dans l'arborescence, et comment les déplacer si nécessaire. Le reste des manipulations est pris en charge par TYPO3 en interne.

Chaque page a un numéro d'identification unique (ID). Il apparaît quand vous placez votre souris sur l'icône d'une page, et est aussi visible dans le formulaire d'édition d'en-tête de page. Par exemple, à la figure précédente, la page « B2C Accueil » a comme ID : 56.

Le rédacteur n'a pas besoin de connaître les numéros d'identification, même s'ils jouent un rôle crucial en arrière-plan. Les liens internes de TYPO3 utilisent ces ID en guise de références. L'avantage est que les liens demeurent même si la page change de position, puisque les ID ne changent jamais.

La structure du site est reprise dans l'arborescence. Le principe de hiérarchie devrait être familier aux utilisateurs de PC. Vu d'une certaine manière, l'arbre tient sur sa tête : le globe (tout en haut de l'arborescence) forme les racines (appelées « Rootline » dans le jargon TYPO3). Les pages, considérées comme les branches, peuvent être ajoutées n'importe où dans l'arborescence. Les pages individuelles, voire même toutes les branches de l'arbre, peuvent être déplacées. Cette structure est identique à celle qui permet de naviguer sur le site, puisqu'elle est à la base de la création des menus.

Les icônes présentes dans l'arborescence indiquent de quel type de page il s'agit. TYPO3 prévoit déjà un certain nombre de types de pages prédéfinis, qui diffèrent par leur comportement et leur fonctionnalité. Les icônes reprennent aussi d'autres caractéristiques, telles que les pages cachées, les droits d'accès et la période d'affichage. Nous expliquons plus en détail ci-après les différents types de pages ainsi que les autres fonctionnalités.

3.5.2 Créer et éditer de nouvelles pages

Référence 195486

Créer de nouvelles pages avec TYPO3 est très simple : sélectionnez le sous-module **Page** ou **Liste** dans le module **Web**. L'aire de navigation présente alors l'arborescence avec toutes les pages auxquelles vous avez accès. Afin de créer une nouvelle page, affichez le menu contextuel en faisant un clic droit sur l'icône de la page à laquelle la nouvelle page se rattachera. Cliquez ensuite sur **Nouveau**. Dans la vue détaillée, des pages ou du contenu peuvent être créés de différentes façons, en fonction des extensions installées et des droits d'accès au système. Selon l'endroit où vous souhaitez insérer votre page, sélectionnez **Page (Dans)** (la page est créée à un niveau inférieur) ou **Page (Après)** (la page est créée au même niveau). Si vous préférez créer la page via l'**assistant**, il vous faut définir sa position en cliquant sur une flèche de positionnement.

Après avoir déterminé la position, le formulaire de création de page s'affiche dans la vue détaillée. La structure de base de cette vue détaillée est la même pour toutes les pages et les éléments de contenu : l'en-tête et le bas de page affichent des icônes pour enregistrer un document (disquette), pour enregistrer et visualiser (disquette avec une loupe), enregistrer et fermer (disquette avec une croix), annuler le dernier changement (croix) ou supprimer l'enregistrement (corbeille avec un losange jaune « attention »). Si l'enregistrement a déjà été modifié, les étapes d'édition peuvent être annulées en cliquant sur l'icône d'annulation (flèche bleue). À l'aide des deux menus déroulants, vous pouvez soit passer d'un enregistrement à un autre, soit entreprendre des actions telles que « enregistrer le document » ou « vider la cache de cette page » [1].

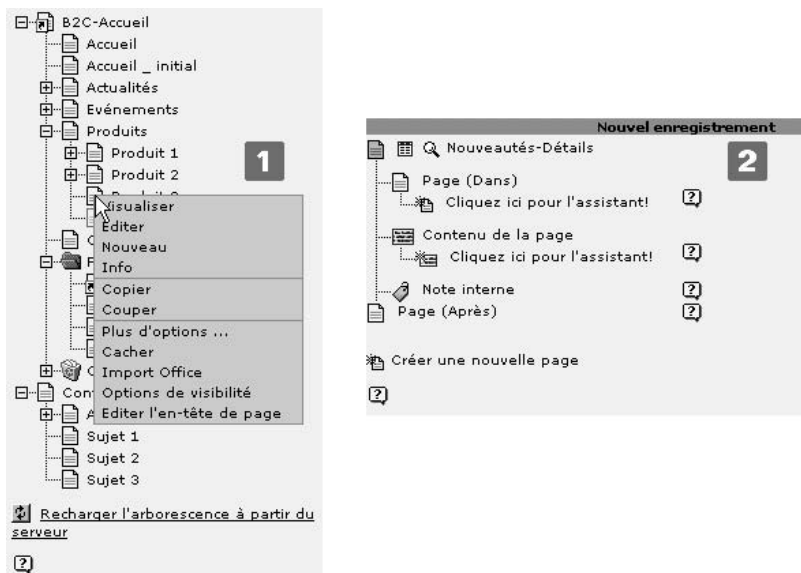


Figure 3.12:
Menu contextuel pour
la page « Produit 3 »
[1] Vue détaillée [2]

Spécifiez le type de page et saisissez les informations obligatoires (marquées d'un carré jaune « attention » si le champ correspondant n'a pas été rempli) dans les champs. Ces derniers sont organisés en blocs ; tous les types de pages ont en commun le bloc déterminant son **type** [2], le **titre de la page** [3], les **paramètres de localisation** [4] et les **options générales** [5].

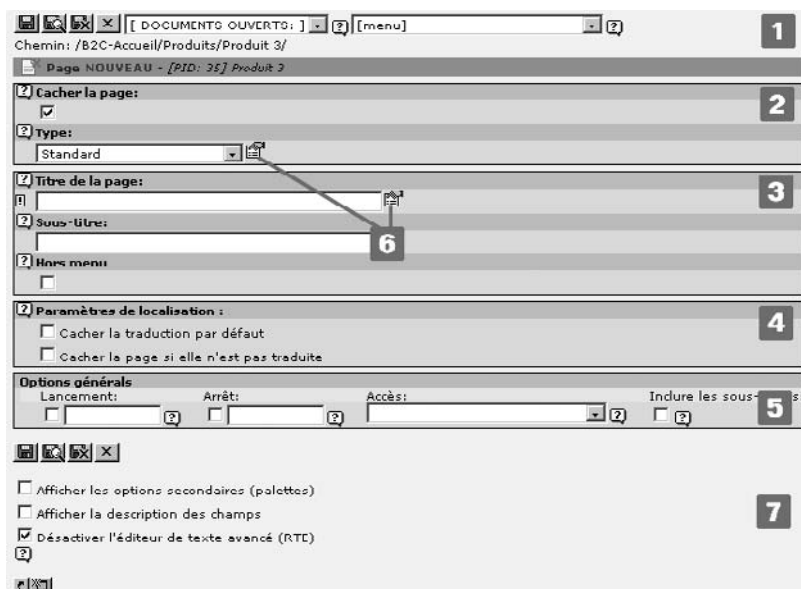


Figure 3.13:
Vue détaillée avec
formulaire de
création d'une
nouvelle page (type
standard). Les
champs TSConfig et
Page de Stockage des
Enregistrements
Généraux ne sont pas
visibles ici pour le
rédacteur

Type

TYPO3 offre une série de types de page prédéfinis, ce qui simplifie le travail quotidien. Ils diffèrent surtout par leur fonctionnalité et par leur apparence dans le frontend. Les types les plus importants pour le rédacteur sont **Standard** et **Avancé**. Le bloc contient aussi le champ **Cacher la page**, qui détermine si une page est visible en ligne ou non. Si une page est invisible dans le frontend, son icône représente une page blanche avec une croix rouge.

Titre de la page

Le **Titre de la page** est le seul champ qui doit obligatoirement être rempli lors de la création d'une nouvelle page. Parmi les options avancées (type de page : **Avancé**), vous pouvez assigner un **Alias** à la page, et ensuite l'appeler en spécifiant l'URL <http://www.votredomaine.com?alias>.

En règle générale, les pages sont « cachées » — elles sont entièrement générées par TYPO3 et le résultat est enregistré dans la base de données. Pourquoi ? Parce que la génération d'une page requiert du temps et des ressources de la part du serveur, alors qu'une page provenant du cache est servie beaucoup plus vite. Afin de maintenir les pages cachées à jour, elles sont normalement régénérées après un délai de 24 heures. Cet intervalle peut être modifié individuellement par l'administrateur pour chaque page. Le rédacteur peut toutefois — en supposant qu'il en ait les droits — les redéfinir, voire décider qu'elles ne soient pas cachées du tout (**sans cache**).

Cette fonction peut être nécessaire pour des pages dont le contenu change d'une minute à l'autre (par exemple : valeurs boursières ou forum).

Paramètres de localisation

On détermine ici si la page est traduite vers la langue par défaut, ou si elle demeure invisible tant qu'elle n'est pas traduite.

Options générales

On y définit le comportement de la page pour laquelle l'affichage est limité dans le temps, ou pour laquelle l'accès est limité à certains utilisateurs. **Lancement** indique le moment où la page sera mise en ligne automatiquement (rendue visible) et **Arrêt** indique quand elle redeviendra invisible.

Les champs de dates indiquent la date sous la forme « jour-mois-année », par exemple « 10-07-2005 ». Une astuce : si vous entrez **d** et passez au champ suivant en appuyant sur la touche de tabulation, la date du jour s'affiche automatiquement dans le bon format. Si vous voulez publier la page par exemple dans 14 jours, il vous suffit de taper **d+14**. Le menu déroulant **Accès** permet de déterminer si l'accès à une page du frontend sera limité à certains groupes seulement, et si une page sera visible ou invisible après l'identification dans le frontend. L'option **Inclure les sous-pages** permet d'appliquer la configuration de la page sélectionnée (**Lancement**, **Arrêt** et **Accès**) à toutes les sous-pages de celle-ci.

Une aide contextuelle associée à chacun des champs est disponible en cliquant sur l'icône représentant un point d'interrogation. L'icône montrant un **formulaire et une main** [6] affiche des champs optionnels dans la barre grise au-dessus de la vue détaillée (non repris dans la figure). En cochant les cases en bas de page [7], vous modifiez l'affichage du formulaire d'édition de la page et de tous les éléments de contenu texte qui s'y rapportent. Une courte description des champs apparaît si la case **Afficher la description des champs** est cochée. La case **Afficher les options secondaires (palettes)** affiche les champs optionnels directement dans la vue détaillée. Vous pouvez dès lors choisir d'avoir une version simple et nette du formulaire, ou d'afficher toutes les options disponibles de façon permanente. La case **Désactiver l'éditeur de texte avancé** permet d'activer ou de désactiver Rich Text Editor pour tous les éléments d'une page utilisant celui-ci comme option.

3.5.3 Différents types de pages

Le type de page sert tout d'abord à déterminer si une page est visible dans le frontend et affichée dans le menu, ou si elle sert seulement de dossier invisible pour entreposer l'information du backend. L'icône de la page dans l'arborescence indique l'usage auquel elle est réservée.

Standard

Le type **Standard** suffit généralement ; il offre une sélection des informations les plus importantes d'une page, comme mentionné plus haut. Si vous passez de ce type de page à un autre comportant des options additionnelles ou différentes, celles qui ne sont pas reprises dans le type **Standard** ne sont pas perdues. Ainsi si vous avez rempli des champs disponibles seulement dans le mode **Avancé** et que vous passez au mode **Standard**, les données resteront disponibles si vous retournez au mode **Avancé**. D'un point de vue technique, les types de page ne sont que des interfaces affichant différentes parties d'une même table de base de données (Pages).

Avancé

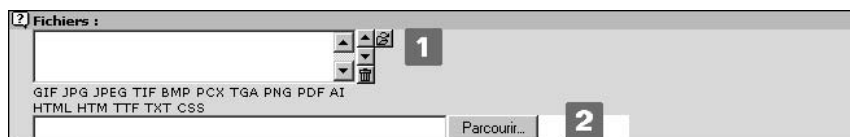
Le type **Avancé** offre des avantages considérables : on y spécifie la méta-information et on y intègre des dossiers ou des plugins. Vous pouvez aussi spécifier un titre différent du titre de navigation pour chaque page, afin que cette dernière apparaisse dans le frontend sous un titre différent de celui qui lui est attribué dans le backend.

Les champs de méta-information tels que **Résumé**, **Mots-clés** et **Description** sont importants pour les moteurs de recherche externes ainsi que pour la fonction de recherche de TYPO3. Il est aussi recommandé de donner aux pages un titre en rapport avec son contenu.

Un certain nombre d'extensions doivent être intégrées directement dans la page via **Contient le plugin** afin de bien fonctionner. Si des **fichiers** doivent être intégrés depuis le gestionnaire de fichiers ou depuis votre PC via le navigateur d'éléments, leur affichage et leur mise en forme dépendent de la configuration du gabarit.

Figure 3.14:

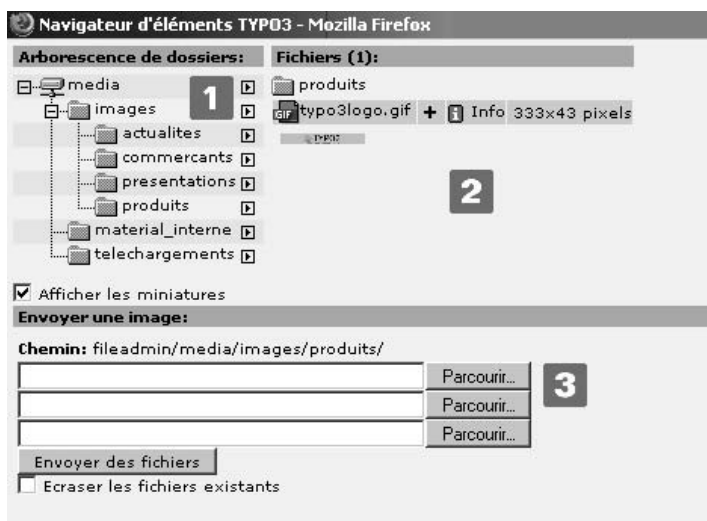
Intégration
d'éléments via le
navigateur
d'éléments [1] ou la
fonction « Parcourir »
[2]



En fonction de la tâche, vous pouvez explorer les pages ou les répertoires du système de fichiers avec le navigateur d'éléments, par exemple pour intégrer des enregistrements, placer des liens ou ajouter directement des fichiers (ex. : images, vidéos, son, version imprimable). Sélectionnez un seul fichier en cliquant sur son titre ou un groupe de fichiers en cliquant sur le symbole « + ».

Figure 3.15:

Navigateur
d'éléments accédant
au système de
fichiers :
arborescence [1],
sélection de fichiers
en cliquant sur le titre
ou le signe « + » pour
une sélection groupée
[2], chargement des
fichiers dans le
système de fichiers
[3]



URL externe

Référence 132947

Le type de page **URL externe** renvoie à une adresse externe de type Internet, FTP ou email. La page en tant que telle n'a pas de contenu, mais est utilisée pour intégrer des liens vers des applications externes dans la structure du menu du frontend. Entrez l'adresse et le type de protocole.

Figure 3.16:

URL externe



Dans certains cas, il peut être intéressant de spécifier une adresse URL interne, pour spécifier dans l'adresse des arguments à passer à un plugin, par exemple pour faire un lien direct vers une page d'actualités en tant que nouvelle page, etc.

Raccourci

Le type de page **Raccourci** (lien) a la même fonction, la seule différence est qu'elle renvoie à une page interne. Dans notre exemple, la page « Accueil » correspond à ce type. On l'utilise quand la page apparaît dans les menus du frontend sans avoir de contenu propre.

Référence 524611

Ce type de page propose deux modes pour définir la page cible :

- 1. Le navigateur d'éléments peut définir un renvoi à une page ou à un élément de contenu spécifique.
- 2. Dans la fonction **Mode raccourci** (options avancées), on peut créer un lien vers la **Première sous-page** ou une **Sous-page aléatoire**.

Dans notre exemple, la première méthode a été utilisée pour lier la page racine « B2C Accueil » à la page « Accueil ».

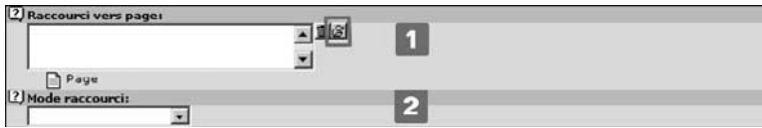


Figure 3.17:
Raccourci via le navigateur d'éléments [1] ou le Mode raccourci [2]

Le navigateur d'éléments s'utilise dans le contexte de l'arborescence de pages à la manière d'un navigateur de fichier. Ce concept utilisateur est présent uniformément dans TYPO3, même dans le Rich Text Editor, afin de maintenir les liens manuellement. Le navigateur d'éléments simplifie la recherche rapide d'une page ou d'éléments de contenu à l'aide de mots-clés.

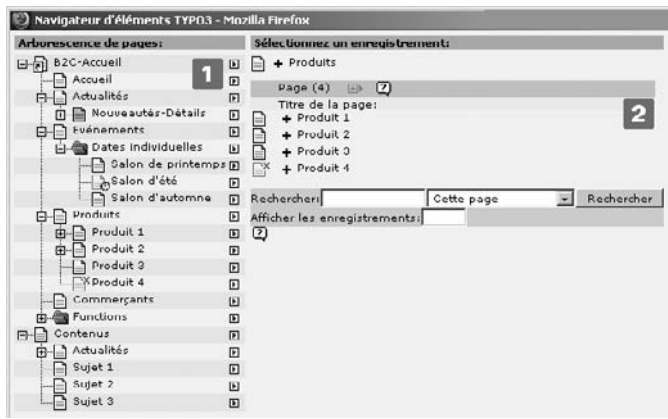


Figure 3.18:
Le navigateur d'éléments fait un lien vers des éléments de contenu ou une autre page via un raccourci : arborescence de pages [1], sélection d'enregistrements à l'aide du titre ou de « + » [2]

Hors menu

Le type de page **Hors menu** est celui qui s'apparente le plus au mode **Avancé** en termes de champs. Ce qui le distingue toutefois est son comportement dans le frontend : la page peut être appelée directement via son numéro d'identification, mais elle n'est pas reprise dans le menu. Dans notre exemple, la page « Nouveautés-Détails » est de ce type.

Référence 157087

Section utilisateur backend

Les pages de la **Section Utilisateur Backend** ne sont visibles dans le frontend que pour les utilisateurs déjà identifiés dans le backend. Ces pages nécessitent elles aussi des droits de lecture, et leur affichage dans le frontend ne peut être appelé que via le backend.

Ne confondez pas le backend avec les pages du frontend réservées à certains visiteurs par un mot de passe. Ces dernières ne sont pas contrôlées via le type de page, mais par le champ **Accès** de chacune des pages et des éléments de contenu.

Point de montage

Le type de page **point de montage** ajoute vers une page cible un raccourci (analogue aux liens symboliques connus des utilisateurs UNIX) à l'endroit où il est inséré. Cela permet, non seulement de définir la page ciblée dans le champ du **point de montage (avancé)** en utilisant le navigateur d'éléments, comme pour une page de type raccourci, mais en plus de garder le format défini pour la page à afficher (même si la page cible utilise un graphisme complètement différent).

Ces pages sont affichées via des paramètres ajoutés à chacune des pages :

`index.php?id=13&MP=8-81`

ce qui correspond à la syntaxe suivante :

`index.php?id=[page appelée]&MP=[page cible]-[page à afficher]`.

De cette manière, vous définissez une réserve de contenus communs dans lesquels différents sites Web peuvent puiser. Le contenu est affiché dans le graphisme propre à chaque site.

Délimiteur

Les pages de type **Délimiteur** servent à structurer les menus. Dans le menu frontend, le titre de la page apparaît mais ne renvoie pas à celle-ci. Vous pouvez donc par exemple créer des en-têtes temporaires dans les menus. Bien sûr, l'affichage doit déjà être défini dans TypoScript par le développeur.

Dossier système

Le **Dossier système** sert essentiellement de réceptacle pour les enregistrements et n'apparaît pas dans le frontend. Il peut servir aux pages et aux éléments de contenu, mais ces dossiers sont surtout prévus pour administrer et structurer les enregistrements ne devant pas apparaître dans le frontend (ex. : catégories pour l'extension des actualités, par lesquelles les messages sont classés, ou la liste de tous les utilisateurs frontend et des groupes). Dans notre exemple, la page « Fonctions » est de type **Dossier système**.

Cependant, on peut aussi créer des sections entières d'une page dans un **Dossier système**, s'il est prévu qu'elles n'utilisent pas la procédure automatique de création de menu. La section « Dates individuelles » contient des pages créées dans notre application B2C, afin d'être sélectionnées dans la zone de contenu.

Corbeille

La **Corbeille** se comporte comme un **Dossier système**, à la différence que son champ d'application est symbolisé par sa propre icône. La **corbeille** de TYPO3 a son propre type de page pour permettre aux utilisateurs backend de déplacer des enregistrements sans les supprimer immédiatement. Les enregistrements ne sont donc pas déplacés automatiquement vers ce dossier lorsqu'ils sont effacés !

Une fois les enregistrements supprimés, vous n'y avez plus accès. S'ils sont conservés dans la **corbeille**, ils peuvent être récupérés et retravaillés à tout moment. Vous pouvez restituer les éléments à leur place dans l'arborescence à l'aide de la fonction annuler ou à l'aide de l'historique.

3.6 Insertion d'éléments de contenu dans TYPO3

Jusqu'à présent, nous avons vu que les pages fournissent l'ossature sur laquelle TYPO3, en tant que système de gestion du contenu, peut intégrer tous les éléments de contenu voulus. A l'intérieur d'une page, les éléments de contenu sont classés par colonnes et par langues. Le concept de colonne permet de regrouper certains éléments de contenu de la page pour ensuite les placer via les gabarits dans des zones précises du frontend. Par exemple, un gabarit peut prévoir une séparation dans le frontend entre une zone principale et une zone secondaire (ex. : pour un message défilant ou une autre application). Chaque page peut contenir un nombre illimité d'éléments de contenu. L'ordre suivant lequel le frontend affiche les éléments de contenu correspond à l'ordre dans lequel ils sont insérés dans les colonnes du backend (module **Page** → **Colonnes**).

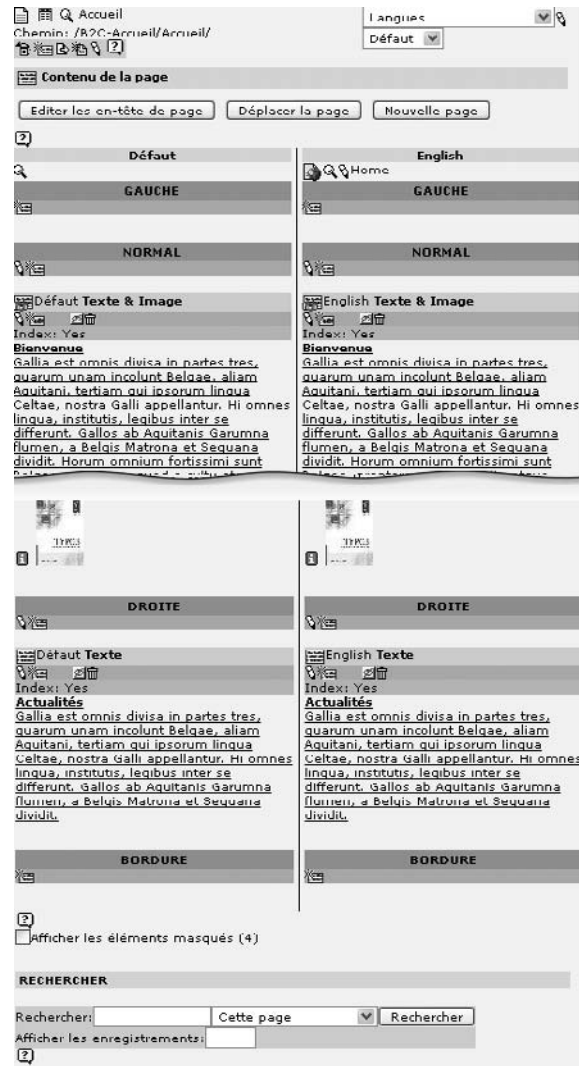


Figure 3.19:
Affichage de deux
zones de contenu
dans le frontend et la
vue détaillée
correspondante du
backend (Module :
Page → Colonnes)

TYPO3 prévoit quatre colonnes par défaut : **Gauche**, **Normal**, **Droite** et **Bordure**. Il est rare que les quatre colonnes soient utilisées d'emblée, mais de nouvelles zones pourraient être ajoutées ou d'anciennes renommées lors de la mise en œuvre du site. De plus, l'administrateur peut déterminer à quelle zone chaque rédacteur a accès, et désactiver les colonnes non utilisées.

Concernant la gestion des langues, TYPO3 présente un mode de fonctionnement similaire : la même page peut supporter du contenu édité dans différentes langues. Dans ce cas, les éléments de contenu sont filtrés en fonction du choix de la langue, et affichés en entier dans le backend, dans le module **Page** → **Langues**. La gestion de sites Web multilingues est reprise plus en détail dans la section 3.10.8.

Figure 3.20:
Vue détaillée du
backend pour la
gestion des langues
(module : Page →
colonnes) ; seules les
colonnes Normal et
Droite et leurs
éléments de contenu
sont repris



3.6.1 Création et édition de nouveaux éléments de contenu

Le formulaire pour la création de nouveaux contenus est fort semblable au formulaire de création de page : cliquez sur **Nouveau** dans le menu contextuel de l'arborescence de pages du module **Web**. Les types de données disponibles apparaissent dans la vue détaillée. Choisissez maintenant **Cliquez ici pour l'assistant !**, et déterminez le type et la position de l'élément de contenu dans la page.

Référence 533217

Figure 3.21:
Assistant pour la
création de contenu :
choix du type [1] et
de la position [2] du
nouveau contenu

La vue détaillée qui apparaît pour la création de nouveaux éléments de contenu ressemble au formulaire de création de page : l'en-tête contient la barre de menu [1], et en pied de page, on peut ajuster les options de visualisation et d'édition pour les sections du formulaire [5]. Si l'éditeur de texte Rich Text Editor est désactivé pour les éléments de contenu **Texte** et **Texte & image**, il n'y a plus de mise en forme du texte dans l'interface d'édition. Ceci peut être très utile si par exemple on veut vérifier les balises insérées par le Rich Text Editor dans le code source.

Les différents types de contenu ont tous dans leur formulaire les mêmes sections pour spécifier le **Type** [3], l'**En-tête** [3] et les **Options générales** [4]. Si les options secondaires (palettes) pour les champs sont désactivées, elles peuvent être affichées en cliquant sur les icônes correspondantes [6].

Figure 3.22:
Formulaire de
création d'éléments
de contenu texte

Type

Dans le formulaire de création de nouveau contenu se trouve une liste reprenant les divers types de contenu (reprise sous le terme technique *CTypes*), allant du simple élément de texte aux éléments multimédias. Lorsqu'on modifie le type de contenu, l'affichage des champs s'adapte automatiquement en fonction des caractéristiques de ce type.

Les types sont présentés individuellement plus bas. Lorsque vous saisissez des données dans les champs de la section **Type** du formulaire, il est important d'associer le contenu à une **Colonne** spécifique — et, pour les applications multilingues, de l'associer à la **Langue** correspondante. Ces champs sont accessibles via les options secondaires. **Avant** et **Après** déterminent l'espace (en pixels) entre cet élément de contenu et respectivement ceux qui le précèdent ou le suivent. Grâce au champ **Cadre**, vous pouvez indenter, cadrer, spécifier une couleur de fond ou souligner votre élément. La case à cocher **Index** détermine si l'élément est repris dans le plan du site de type **Index des sections (Contenu w/Index coché)** — par exemple lorsque le texte est trop long pour être affiché dans son ensemble. **Vers le haut** ajoute un petit lien de retour vers le haut de la page lorsqu'il est activé. Il vous permet de retourner vers le haut de la page en un seul clic au lieu de recourir à la barre de défilement.

Figure 3.23:
Section **Type** du
formulaire de
création de contenu

En-tête

Lorsqu'il est affiché, l'en-tête est généralement mis en relief par rapport au reste de l'élément de contenu en recourant à des caractères gras. Le champ **Type** offre plusieurs possibilités de mise en forme pour l'en-tête.

Si vous ne voulez pas que l'en-tête s'affiche sur votre site, activez l'option **Caché** du champ **Type**. Les autres champs permettent d'aligner l'en-tête (gauche, centre, droite) ou d'ajouter un lien ou une date. Certains types d'éléments de contenu tels qu'**Insérer enregistrements** ou **HTML**, ne présentent pas d'en-tête ni d'options secondaires s'y rattachant dans leur configuration de base. Ces types de contenu ne présentent que le champ **Titre**.



Figure 3.24:
Section En-tête du
formulaire

Options générales

À l'instar des **Options générales** pour l'édition des pages, les éléments de contenu peuvent eux aussi être rendus invisibles grâce à la fonction **Cacher**. Les champs **Lancement** et **Arrêt** déterminent une publication limitée dans le temps. Avec **Accès**, il est possible de restreindre la visualisation à certains groupes d'utilisateurs frontend.

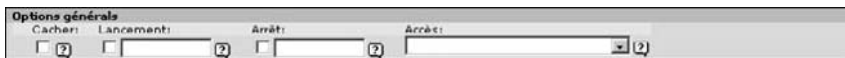


Figure 3.25:
Section Options
générales du
formulaire

3.6.2 Types de contenu

Même dans sa version simplifiée, TYPO3 offre toute une gamme de types de contenu couvrant les cas typiques d'insertion sur une page HTML. Vous pouvez ajouter de nouveaux types de contenu en développant des extensions. Les pages suivantes présentent les principaux types accompagnés d'exemples. Les points plus particuliers tels que la mise en forme personnalisée de types de contenu pour l'affichage dans le frontend sont repris au chapitre 6.



Figure 3.26:
Types de contenu
dans le backend

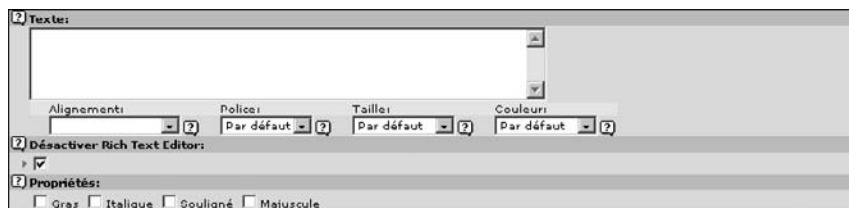
Titre

Cet élément de contenu est rarement utilisé : en effet, il est déjà intégré dans la plupart des autres éléments de contenu. Il ne diffère de ces derniers que par l'ajout d'une option pour l'insertion d'un sous-titre et une variété de mises en page.

Texte

Le type de contenu **Texte** est très fréquemment utilisé pour des passages de texte. On insère l'élément dans le champ **Texte** et on l'ajuste à l'aide des cases à cocher **Gras**, **Italique**, **Souligné** et **Majuscule** ou des options secondaires **Alignement**, **Police**, **Taille** et **Couleur**. La mise en forme s'applique ici au texte dans son ensemble : si vous souhaitez seulement réarranger des passages isolés (ex. : ajouter des liens ou des images), utilisez le Rich Text Editor.

Figure 3.27:
Type de contenu
Texte : section
d'insertion du
formulaire



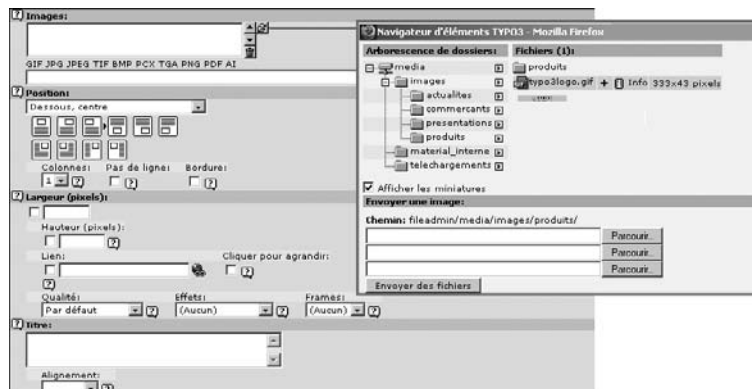
Image

Référence 550518

Les images sont normalement insérées dans une page via les éléments **Image** ou **Texte & image**. Pour y parvenir, sélectionnez les images dans la gestion des médias du système à l'aide du navigateur d'éléments, ou chargez-les directement à partir de votre réseau local à l'aide du bouton **Parcourir**. La fonction **Alignement** détermine sa position (aligné à gauche, à droite ou centré). Le type de contenu **Texte & image** définit directement la position de l'image par rapport à celle du texte.

Si vous désirez utiliser plusieurs images, vous pouvez les disposer dans un tableau invisible à l'aide de l'option **Colonnes**. La hauteur des lignes du tableau s'ajuste à la taille de l'image la plus grande qu'elles contiennent. Cet ajustement automatique peut être désactivé en cochant la case **Pas de ligne**. Un tableau sera alors inséré pour chaque ligne, dont les proportions s'ajustent à toutes les images qu'il contient. Vous pouvez aussi afficher un cadre autour de l'image en activant la case à cocher **Bordure** (pour accentuer le contraste par exemple).

Figure 3.28:
Type de contenu
Image : intégration
d'éléments à partir du
gestionnaire de
fichiers via le
navigateur
d'éléments



TYPO3 peut intégrer des images de n'importe quel format et les éditer à l'aide de ses propres fonctions graphiques. Vous pouvez ainsi en modifier la taille, la qualité et le format.

Vous pouvez déterminer la taille de l'image avec les champs **Largeur** et **Hauteur** (en pixels). **Qualité** permet en plus de définir son format (GIF, PNG, JPG), la profondeur de ses couleurs ou son taux de compression. Vous avez même la possibilité de retravailler les images et leurs **effets** après une première sauvegarde. Vous pouvez ajuster le contraste, éclaircir, aiguïser, normaliser, convertir en nuances de gris et leur faire subir une rotation de 90° ou de 180°.

Des cadres peuvent être créés pour les images simplement en utilisant les balises HTML, mais on peut aussi créer des cadres à partir d'images (cf. figure suivante). Ces images de cadres sont disponibles dans le menu déroulant **Frames**. L'installation par défaut propose deux séries d'exemples, « Artists » et « Darkroom » dans le répertoire `tslib/media/frames/`. Ils peuvent bien sûr être personnalisés lorsque le développeur configure le système (cf. `styles.content.img-Frames` dans le gabarit standard `styles.content (default)`).



Figure 3.29:
Ajout de cadres à
l'aide de l'image de
cadre 2 de la série
« darkroom »

Dans le frontend, l'image peut renvoyer à une adresse URL spécifiée dans le champ **Lien**. Si vous cochez l'option **Cliquer pour agrandir**, l'image apparaît à sa taille initiale dans une nouvelle fenêtre lorsqu'on clique sur l'image miniature.

Le **Titre** apparaît normalement sous l'image, mais il peut être déplacé (aligné à droite, à gauche ou centré) à l'aide de l'option **Alignement**. Lors de la saisie de titres pour plusieurs images, le texte du titre doit être introduit dans le même ordre que les images, à chaque fois sur une ligne séparée.

Note : Pour associer différents titres à différentes images, la ligne de code suivante doit être rajoutée dans le gabarit : `styles.content.imgtext.captionSplit = 1`

Sinon, les titres seront affichés en un seul bloc pour toutes les images.

Texte & image

L'élément de contenu **Texte & image** est une combinaison des types **Texte** et **Image** et comporte des options des deux éléments. Il offre en plus la possibilité de déterminer la position de l'image par rapport au texte (y compris l'habillage) à l'aide de l'option **Position**.

Liste à puces

Référence 552631 La **Liste à puces** constitue un type de contenu idéal pour les listes. Chaque ligne entrée dans le champ texte apparaît comme l'élément d'une liste dans le frontend. Les retours à la ligne dans un même élément sont indiqués avec l'expression HTML `<br>`.

Quatre types de **Mise en page** sont disponibles, qui par défaut utilisent des icônes, des nombres ou des coches afin de distinguer visuellement les éléments. Ils peuvent bien sûr être redéfinis par le développeur.

Comme pour le type de contenu **Texte**, le contenu peut aussi être mis en forme. Les options disponibles sont **Gras**, **Italique**, **Souligné** et **Majuscule**, ainsi qu'**Alignement**, **Police**, **Taille** et **Couleur**.

Tableau

Référence 727880 Pour publier un tableau HTML, utilisez le type de contenu **Tableau**. Ici aussi, chaque ligne de contenu représente une ligne dans le tableau HTML. Les colonnes dans une même ligne doivent être marquées par le caractère « | » (le symbole « tube »⁵).

Voici l'exemple d'un tableau de deux lignes et quatre colonnes (le contenu de la première colonne est un espace vide) :

```
| Colonne 2 | Colonne 3 | Colonne 4
Ligne 2 | Apud Helvetios... | Ea res... | Damnatum...
```

Par facilité, le texte de la deuxième ligne est ici abrégé. En voici le résultat dans le frontend :

Figure 3.30:
Exemple de tableau
utilisant la mise en
page standard 3

Tableau			
	Colonne 2	Colonne 3	Colonne 4
Ligne 2	Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit.	Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate provinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent important, proximique sunt Germanis, qui trans Rhenum incolunt, quibuscum continenter bellum gerunt.	Qua de causa Helvetii quoque reliquos Gallos virtute praecedunt, quod fere quotidianis proeliis cum Germanis contendunt, cum aut suis finibus eos prohibent aut ipsi in eorum finibus bellum gerunt. Eorum una, pars, quam Gallos obtinere dictum est, initium capit a flumine Rhodano, continetur Garumna flumine.

L'apparence du tableau peut être modifiée dans la section **Mise en page**. Quatre variantes sont disponibles dans la configuration standard, qui permettent d'ajouter de la couleur, par exemple, aux titres de colonnes et de lignes. Il existe aussi des options pour définir la **Couleur de fond**, la largeur de la **Bordure**, l'**Espacement entre cellules** et l'**Espacement inter cellules**. Le nombre de colonnes est soit détecté automatiquement, soit il est défini manuellement via le menu déroulant **Colonnes du frame**. Toutefois, si vous avez limité le nombre de colonnes à trois par exemple, le contenu correspondant à des colonnes dépassant le nombre de trois sera ignoré.

⁵aussi appelé « pipe » en anglais

La mise en forme du contenu utilise les mêmes fonctions que pour le type **Texte**, via les options standards et secondaires.

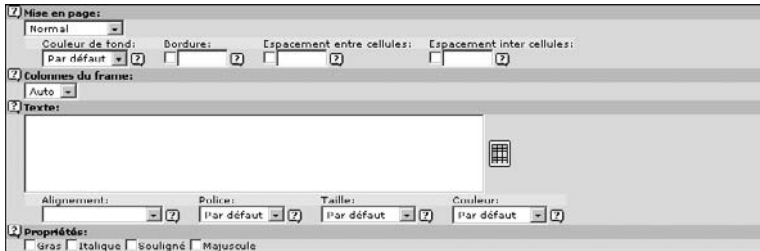


Figure 3.31:
Formulaire pour le
type de contenu
Tableau

Après avoir enregistré un **Tableau** une première fois, un assistant est disponible afin de simplifier les modifications ultérieures. Vous l'activez en cliquant sur l'icône se trouvant à la droite du champ texte. Une nouvelle fenêtre s'ouvre alors, dans laquelle vous pouvez éditer, effacer et ajouter du contenu au tableau à l'aide d'une interface graphique très simple d'utilisation. Vous pouvez élargir les champs individuels qui vous semblent trop petits, en cas de longs extraits de texte, en désactivant la case à cocher **Petits champs**.

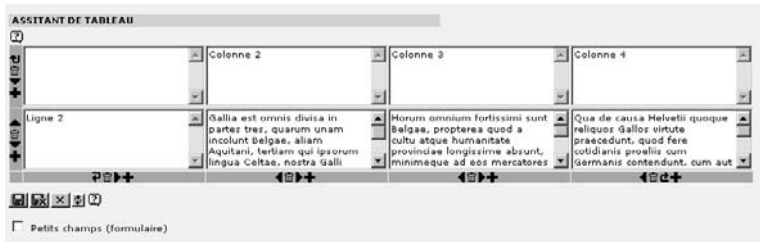


Figure 3.32:
Assistant à la
création de contenu
dans un tableau

Lien vers fichier

L'élément de contenu **Lien vers fichier** sert à afficher divers types de fichiers sous forme de tableau dans le frontend ; en fonction du type de média, un visiteur peut ouvrir le lien ou télécharger de l'information provenant du système de fichiers.

Référence 976910

	Demarrer_avec_Typo3.pdf	219 K
	step_tut.zip	14.6 K
	t3_livre_couverture.png	88 K
	typo3logo.gif	2.1 K
	Manuel_du_redacteur.pdf	634 K

Figure 3.33:
Exemple du type de
contenu **Lien vers
fichier** appliqué à
différents types de
médias

Les fichiers peuvent être facilement intégrés à l'aide du navigateur d'éléments, le bouton **Parcourir**, ou **Lire depuis le chemin**. Le rédacteur sélectionne le navigateur d'éléments dans la section **Fichiers**, ouvre les dossiers mis à sa disposition, et choisit les fichiers qui l'intéressent.

Il est aussi possible d'envoyer des fichiers sur le serveur à partir du PC local à l'aide du bouton **Parcourir**. Cette méthode toutefois ne permet pas de voir les fichiers dans le navigateur d'éléments, car ils ne sont pas enregistrés dans le système sous le répertoire `fileadmin/`, mais sous `uploads/media/`.

Pour afficher tous les fichiers contenus dans ce dossier, vous devez spécifier le chemin dans **Lire depuis le chemin**.

Figure 3.34:
Formulaire pour le
type de contenu Lien
vers fichier

The screenshot shows the 'Lien vers fichier' (Link to file) configuration form in TYPO3. It is divided into several sections:

- fichiers:** A section for selecting a file, featuring a file browser icon and a 'Parcourir...' (Browse...) button.
- Mise en page:** A section for configuring the display of the link, including a 'Normal' dropdown, a 'Couleur de fond' (Background color) field with a 'Par défaut' (Default) button, and three input fields for 'Bordure' (Border), 'Espace entre cellules' (Space between cells), and 'Espace inter cellules' (Inter-cell space).
- Afficher taille du fichier:** A checkbox to display the file size.
- Descriptions:** A text area for adding a description to the link.

Ces liens vers les fichiers peuvent s'afficher de quatre façons différentes dans le frontend (via l'option **Mise en page**). Ces quatre modèles sont préconfigurés dans TYPOScript, et par défaut, ils affichent le titre du fichier avec l'icône associée au type de fichier ou l'image en prévisualisation. Grâce à l'affichage sous forme de tableau dans le frontend, la couleur de fond, la bordure, l'espace entre cellules et l'espace inter-cellules peuvent être paramétrés individuellement.

De l'information peut être ajoutée au titre du fichier en cochant la case **Afficher taille du fichier** et en remplissant le champ **Descriptions**. Tout comme pour les titres d'images, la description des fichiers est insérée à chaque fois sur une ligne séparée. De plus, une ligne de code doit figurer dans le TYPOScript du gabarit (`styles.content.uploads.descriptionWrap`).

Multimédia

Référence 986146

À l'aide du type **Multimédia**, vous pouvez intégrer des fichiers de format TXT, HTML, HTM, CLASS, SWF, SWA, DCR, WAV, AVI, AU, MOV, ASF, MPG, WMV, ou MP3 en tant que contenu d'une page. Vous les sélectionnez dans la section **Fichier**, soit à l'aide du navigateur d'éléments, soit directement à partir de l'environnement de travail local en les plaçant dans le répertoire interne du système `uploads/media/`.

Dans le second cas, la règle s'applique à nouveau : un fichier téléchargé vers le serveur de cette façon n'apparaîtra pas dans le navigateur d'éléments, mais n'est lié qu'à cet élément de contenu.

Les paramètres nécessaires à la lecture de vidéos, de fichiers audio ou d'animations Flash peuvent être saisis dans le champ **Paramètres**. Nous avons ici comme exemples les paramètres pour une animation Flash :

```
WIDTH=70
HEIGHT=55
LOOP=true
QUALITY=high
BGCOLOR=#FFFFFF
```

```
TYPE="application/x-shockwave-flash"
PLUGINSOURCE=http://www.macromedia.com/shockwave/download/index.cgi?1_Prod_Version=
ShockwaveFlash;
```

Formulaire

L'élément de contenu **Formulaire** permet de créer des formulaires email. Un assistant est à votre disposition. Il utilise la notation suivante : dans la section **Configuration**, chaque ligne correspond soit à un champ du formulaire à créer, soit à une fonction.

Référence 404318

```
Nom: | *Nom=input,40 | [Veuillez saisir votre nom]
Email: | *email=input,40
Votre adresse: | Adresse=textarea,34,4
Mon appréciation de TYPO3: | Vote=radio | Super=super, Génial=génial,
Inégalable=inégalable
| formtype_mail=submit | Soumettre!
| html_enabled=hidden | 1
| subject=hidden | Mon appréciation de TYPO3
```

Le formulaire email est affiché sous forme de tableau. Le symbole « | » sert ici aussi à séparer le titre (description ou commentaire) du type de champ de formulaire et des détails de la configuration.

Élément	Première valeur (Titre)	Deuxième valeur (type de champ)	Troisième valeur (détails de configuration)
zone de texte	Label	[*=obligatoire][nom du champ =] textarea [colonnes, lignes, "wrap=[p. ex."OFF"]"]	[donnée par défaut]
champ texte	Label	[*=obligatoire][nom du champ =] input [taille max.]	[donnée par défaut]
mot de passe	Label	[*=obligatoire][nom du champ =] input [,taille,max]	[donnée par défaut]
fichier	Label	[*=obligatoire][nom du champ (*1)=] file [,taille]	
case à cocher	Label	[*=obligatoire][nom du champ =]check	[coché=1]
liste déroulante	Label	[*=obligatoire][nom du champ =]select [,taille (int/"auto"),"m"=multiple]	Label [=valeur] ,...
bouton radio	Label	[*=obligatoire][nom du champ =]radio	Label [=valeur] ,...
champ caché		[nom du champ =]hidden	Valeur
envoyer	Label	[nom du champ =]submit	Légende

Tableau 3.1:
Aperçu de la configuration du formulaire

Le champ **Aller à la page** spécifie sur quelle page se retrouve le visiteur après avoir soumis le formulaire. Cette page de confirmation doit certainement apparaître pour informer le visiteur que son email a bien été envoyé.

Il est primordial de spécifier une adresse email valide dans le champ **E-mail destinataire**. On peut saisir plusieurs adresses en les séparant par des virgules.

Une fois que vous avez enregistré les données, l'assistant mentionné plus haut peut être appelé en cliquant sur l'icône à droite du champ **Configuration**. Il sert à créer et à configurer plusieurs types de champs de formulaires (n'oubliez pas d'enregistrer régulièrement !). Dans le bas de page de l'assistant, une case à cocher donne la possibilité d'activer le mode HTML pour les emails.

L'assistant se charge ensuite de traduire vos entrées dans la syntaxe propre au formulaire de TYP03.

Figure 3.35:
Assistant pour la
création de
formulaire et
affichage dans le
frontend

ASSISTANT DE FORMULAIRE			
Prévisualisation de l'éléments		Type de l'éléments	Configuration détaillée
  	Nom :	Type: Champ texte Etiquette: Nom : Obligatoire: ☒	Champ: Nom Taille: 40 Max: Valeur: (veuillez entrer votre nom)
	Email	Type: Champ texte Etiquette: Email Obligatoire: ☒	Champ: Email Taille: 40 Max: Valeur:
		Adresse	Type: Zone de texte Etiquette: Adresse Obligatoire: ☐
  	Mon appréciation de TYP03		Type: Boutons 'radio' Etiquette: Mon appréciation de TYP03 Obligatoire: ☐
		Configuration spéciale de courrier électronique: 	
Etiquette du bouton d'envoi: Soumettre			
Mode HTML activé: ☒			
Sujet: Mon appréciation de TYP03			
Email du destinataire: t3live@typo3.org			

Formulaire

Nom :

Email :

Adresse :

☐ Super
☐ Brillant
☐ Complet

Recherche

Un formulaire de recherche simple mais puissant est déjà prévu dans la version de base de TYPO3. Il permet à l'utilisateur frontend de rechercher des éléments de contenu, des en-têtes ou des mots-clés dans le site. Ce formulaire est créé à l'aide du type de contenu **Recherche**. Si vous désirez que les résultats de la recherche soient affichés sur une nouvelle page plutôt que sur la page du formulaire de recherche, vous devez spécifier une page dans le champ **Envoyer à la page** – qui doit déjà avoir été créée (de préférence une page de type **Hors menu**).

Référence 481291

Recherche

Echelle de recherche: 1-2 de 2

Commerçants

Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt...

Accueil

Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. Hi omnes lingua, institutis, legibus inter se differunt...

Chercher:

dans

Figure 3.36:
Formulaire de
recherche avec les
résultats affichés sur
la même page

Il existe, en plus de la recherche standard, une version plus puissante, l'**Indexed Search Extension**, qui peut être installée et utilisée au besoin.

Référence 724882

Identifiant

Le formulaire d'identification permet au visiteur qui en a les droits d'accéder aux pages et aux éléments de contenu du frontend qui lui sont réservées et qui restent donc invisibles aux autres visiteurs. Le formulaire d'identification est créé à l'aide du type de contenu **Identifiant**. On peut déterminer une page sur laquelle l'utilisateur sera renvoyé après s'être identifié à l'aide du navigateur d'éléments. Tout cela implique bien sûr que les utilisateurs et les groupes du frontend aient été inscrits correctement et que le formulaire d'identification « sache » où les données utilisateur ont été enregistrées. Dans notre exemple, B2B et B2E, les données utilisateur sont situées dans le dossier système **FEUtilisateurs**.

Référence 673847

Les administrateurs peuvent ajuster ce paramètre en utilisant TypoScript (`fe_adminLib.pid = [uid]`) ou en spécifiant une seconde page de renvoi comportant les données utilisateur. Le chapitre 4.11 donne plus d'informations sur les utilisateurs frontend et leur configuration dans le système.

Figure 3.37:
Formulaire
d'identification

Textbox

Le type de contenu **Textbox** permet au rédacteur d'utiliser une mise en page prédéfinie pour introduire du texte. La configuration par défaut prévoit l'insertion du contenu dans un tableau de deux colonnes, la colonne de gauche pour l'affichage d'images avec différents cadres, et la colonne de droite pour le texte, indenté de 30 pixels. La mise en page (chacune spécifiée par le développeur) est sélectionnée dans le menu déroulant **Type de Textbox**.

Figure 3.38:
Section d'insertion de
contenu pour le type
Textbox

L'élément suivant a été créé à l'aide de la configuration par défaut (cf. content (default)) :

Figure 3.39:
Exemple d'affichage
dans le frontend pour
le type de contenu
Textbox



Textbox

Ea res est Helvetis per indicium enuntiata. Moribus suis Orgetoricem ex vinculis causam dicere coegerunt, damnatum poenam sequi oportebat, ut igni cremaretur. Die constituta causae dictionis Orgetorix ad iudicium omnem suam familiam, ad hominum milia decem, undique coegit, et omnes clientes obaeratosque suos, quorum magnum numerum habebat, eodem conduxit, per eos ne causam dicere se crepuit. Cum civitas ob eam rem incitata armis ius suum exequi conaretur multitudinemque hominum ex agris magistratus cogerent, Orgetorix mortuus est, neque abest suspicio, ut Helvetii arbitrantur, quin ipse sibi mortem consciverit.

Menu/Plan site

Référence 528240

Le **Menu/Plan site** contient divers éléments de contenu facilitant la navigation. On peut y créer des menus de navigation, une description du contenu des pages, ou afficher un plan du site Web en entier. Sélectionnez la fonctionnalité via **Type de menu**, et choisissez un ou plusieurs points d'entrée pour le menu via **Point d'entrée**. Si aucun point d'entrée n'est spécifié, la page à laquelle se trouve le menu est utilisée par défaut.

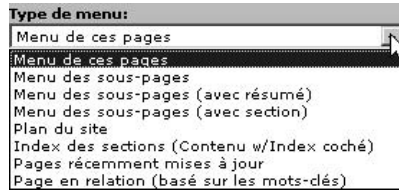


Figure 3.40:
Différents menus
pour le type de
contenu Menu/Plan
du site

Les types de menus prédéfinis sont les suivants :

Menu de ces pages

Crée une liste de liens vers les titres des pages sélectionnées comme points de départ.

Menu des sous-pages

Crée — en fonction de la page sélectionnée comme point de départ — la liste des pages situées à un niveau supérieur ou inférieur.

Menu des sous-pages (avec résumé)

Affiche le titre et les détails à propos du contenu des sous-pages. Pour ce, il faut avoir entré les détails concernant chacune des pages listées dans le champ approprié des options secondaires de la page.

Menu des sous-pages (avec section)

Ce menu affiche le titre de la page et les en-têtes des éléments de contenu dont l'option **Index** est activée.

Plan du site

Dans la configuration par défaut, ce type de menu crée un aperçu des titres de pages organisés en arborescence, dont le niveau de ramification peut être défini ; la mise en page se détermine via TypoScript (styles.sitemap.text).

Index des sections (Contenu w/Index coché)

Ce type de menu est utile pour donner un aperçu du contenu de longues pages. Les en-têtes de tous les éléments de contenu dont l'option **Index** est activée y sont listés.

Pages récemment mises à jour

Liste des dix dernières pages qui ont été modifiées au cours des sept derniers jours, et qui ne sont pas exclues de la liste par l'option **Sans recherche**.

Page en relation (basé sur les mots-clés)

Affiche tous les titres des pages qui contiennent les mêmes mots-clés ; les pages peuvent aussi être exclues de cette liste si l'option **Sans recherche** est activée.

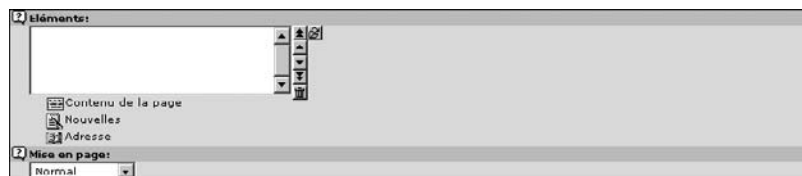
Insérer enregistrements

Le type de contenu **Insérer enregistrements** permet d'utiliser certains éléments de contenu plusieurs fois sur un même site. De cette façon, on ne copie pas l'enregistrement, mais on crée simplement une référence vers l'original. Si l'original est mis à jour, les références reprendront aussi les dernières modifications.

Référence 133467

Le champ **Éléments** et le navigateur d'éléments permettent de créer des relations vers tous les types de contenu classiques et vers les enregistrements des extensions (ex. : un enregistrement d'adresse). La personnalisation de l'affichage des enregistrements dans le frontend via le champ **Mise en page** est définie par le TypoScript du gabarit de la page.

Figure 3.41:
Type de contenu
Insérer
enregistrements



Insérer un plugin

Les extensions du frontend (plugin) organisent des structures de contenu plus complexes sous la forme de catégories, de listes ou de vues détaillées. Elles enrichissent la gamme des fonctionnalités disponibles, par exemple pour une boutique en ligne ou un moteur de recherche.

Pour faire en sorte qu'un plugin soit affiché correctement dans une page, l'extension correspondante doit d'abord avoir été installée et configurée correctement par l'administrateur. Son installation est expliquée plus en détail au chapitre 6.

Script

Le type de contenu **Script** permet d'intégrer vos propres fonctions dans le site Web. À cette fin, le développeur doit d'abord définir la fonction dans le gabarit, et ensuite, celle-ci peut être appelée via le champ **CODE**.

Exemple : la configuration suivante devrait être intégrée dans le gabarit avec **sectionscript** :

```
tt_content.script {
    key.field = select_key
    sectionscript = PHP_SCRIPT
    sectionscript.file = fileadmin/editor/script/section.inc
}
```

La validité du script à travers l'arborescence des pages est spécifiée par les champs **Point d'entrée** et **Récursion**. D'autres **paramètres** peuvent être passés comme arguments aux scripts dans les champs correspondants⁶.

Séparation

Le type de contenu **Séparation** sert simplement à structurer le contenu d'une page très chargée dans le backend — et par le fait même dans le frontend à l'aide de la configuration appropriée (telle que la balise `<hr />`). Le contenu peut dès lors être divisé en blocs.

⁶Cette mention est faite dans un souci d'exhaustivité. En effet, les rédacteurs n'entrent normalement pas en contact avec ce type de contenu pour des raisons techniques ainsi que de sécurité.

HTML

Ce type de contenu permet l'insertion de lignes de code HTML à l'état pur. Il est conseillé de le désactiver pour les rédacteurs, en grande partie pour une question de sécurité.

3.7 Ressources dans TYPO3

3.7.1 Gestion des ressources dans l'arborescence des fichiers

Par *ressources*, nous entendons les médias et les fichiers de toute sorte qui sont nécessaires à la conception d'un site Web. Il s'agit d'images, de son, de vidéos et de documents de tous types, mais aussi des documents de définition tels que des feuilles de style ou des gabarits HTML. Ils sont enregistrés sur le serveur Web dans le répertoire `fileadmin/` et disponibles dans TYPO3 pour être traités et intégrés.

Référence 599239

Il est recommandé de créer une structure de répertoire la plus claire et la plus logique possible, afin que chacun s'y retrouve facilement. En particulier, les droits d'accès des différents utilisateurs et groupes d'utilisateurs doivent être pris en compte. La tâche d'organiser une structure et d'accorder aux différents rédacteurs des droits d'accès à des sections spécifiques du système de fichiers revient à l'administrateur. En fonction de la taille de l'application, il est recommandé de grouper les fichiers selon le type de média et de structurer des dossiers de façon similaire à l'arborescence, parce que les rédacteurs comprennent facilement et rapidement une telle structure.



Figure 3.42:
Administration de
références via le
module Fichier →
Fichiers : accès
restreint au point de
montage media

Lorsque les droits sont accordés, il est important de savoir quels utilisateurs ont reçu des droits pour administrer les branches de l'arborescence de répertoires auxquels ils ont accès. La quantité de ressources grandit en général très rapidement. C'est pourquoi il est utile de les conserver dans des répertoires de taille raisonnable. Pour finir, un certain nombre de rédacteurs doivent avoir la permission de **créer, renommer, copier** et/ou **déplacer** des répertoires.

Les ressources sont gérées dans le backend de TYPO3 dans le module **Fichier → Fichiers**. L'arborescence apparaissant dans l'aire de navigation [1] montre la structure prévue pour le rédacteur. En cliquant sur le nom du dossier, le contenu de chacun des dossiers s'affiche dans

la vue détaillée [2]. Un menu contextuel existe aussi dans les deux sections de l'interface (aire de navigation et vue détaillée). Au bas de la vue détaillée, vous pouvez choisir d'afficher le presse-papiers ou les vignettes des images en cochant les cases appropriées [3]. Le presse-papiers est toujours disponible dans TYPO3 et vous permet de déplacer, de copier ou d'effacer des fichiers ou des éléments de contenu. Il permet même de travailler tout un lot de fichiers à la fois. Nous y reviendrons plus en détail à la section 3.10.

Le menu contextuel de l'arborescence de répertoires comporte les actions **Nouveau** et **Envoyer des fichiers**. Vous pouvez créer plusieurs dossiers en même temps à l'aide de **Nouveau** ou créer des fichiers texte dans les formats TXT, HTML, HTM, CSS, INC, PHP, PHP3, TMPL, JS, ou SQL — mais certains formats peuvent toutefois être désactivés pour des raisons de sécurité. **Envoyer des fichiers** permet d'envoyer jusqu'à dix fichiers simultanément, du réseau local au système de fichiers. Lors de cette action, si vous le spécifiez, les fichiers existants peuvent être remplacés, ou alors les fichiers ayant le même nom sont renommés par défaut avec un numéro séquentiel.

Figure 3.43:
Envoi de fichiers
locaux vers le
système de fichiers de
TYPO3



3.7.2 Insérer des ressources dans une application

Référence 525663

Il n'est pas nécessaire d'être dans le module **Fichiers** pour envoyer des fichiers. Les fichiers peuvent aussi être copiés vers le serveur Web au moment de la création d'une page. Le navigateur d'éléments permet de sélectionner des fichiers déjà existants, mais aussi de télécharger des fichiers [3].

Il existe généralement deux façons de manipuler des ressources fichier : les sélectionner à partir du système de fichiers avec le navigateur d'éléments [1], ou directement avec le bouton **Parcourir** [2] (envoi à partir du poste de travail local).

En fonction de la façon dont ils ont été copiés sur le serveur, les fichiers sont enregistrés à des endroits différents, ce qui détermine leur disponibilité par la suite. Un fichier chargé via le navigateur d'éléments vers le système de fichiers est disponible pour plusieurs usages sur un même site. Ce n'est pas le cas pour un envoi direct via **Parcourir** [2], parce que le fichier est alors enregistré dans un répertoire système interne (`uploads/`), inaccessible au rédacteur.

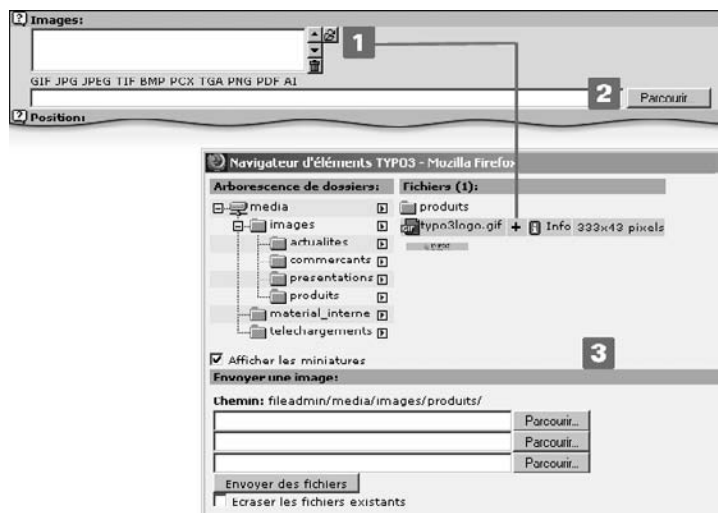


Figure 3.44:
Envoi d'un ensemble
de fichiers en tant
que ressources vers le
système de fichiers
[1] ou séparément à
l'aide de la fonction
Parcourir [2] ;
illustration des
étapes dans le
contexte de
l'insertion d'une
image dans un
élément de contenu
Image

Si le fichier est intégré plusieurs fois via le navigateur d'éléments, TYPO3 détecte automatiquement les copies d'une même image à l'aide d'une combinaison unique de chiffres (ex. : livre.gif, livre_01.gif etc.). Le Rich Text Editor agit de la même manière lorsqu'il intègre des ressources graphiques. Le fichier original (ex. : *.bmp) aussi bien que le format cible (ex. : *.jpg) sont chacun enregistrés et référencés en tant qu'objets uniques. Il est ainsi impossible de confondre deux fichiers, puisque des données uniques sont assignées à chaque enregistrement, et que ces données demeurent, même si les fichiers d'origine sont effacés du système de fichiers. On peut désactiver cette fonction dans TypoScript (cf. chapitre 4.8).

RTEmagicP_t3-livre.bmp	333 KB	BMP File
RTEmagicC_t3-livre.bmp.jpg	14 KB	JPG File
RTEmagicP_t3-livre_01.bmp	333 KB	BMP File
RTEmagicC_t3-livre_01.bmp.jpg	14 KB	JPG File
RTEmagicP_t3-livre_02.bmp	333 KB	BMP File
RTEmagicC_t3-livre_02.bmp.jpg	14 KB	JPG File

Figure 3.45:
t3-livre.bmp a été
intégré trois fois dans
différents éléments
de contenu

3.8 Édition frontend

En mode d'édition frontend, les pages et leur contenu sont directement modifiés à partir du site Web lui-même (« éditez tout en surfant »).

Référence 615520

Pour beaucoup d'utilisateurs qui, en raison des tâches qui leur sont assignées, ne travaillent qu'occasionnellement sur le système, il s'agit à coup sûr de la méthode la plus simple, rapide, intuitive et pratique. Le mode d'édition frontend est aussi bien adapté à un travail spécifique tel que la correction finale d'un texte. L'administrateur a la possibilité de restreindre l'accès à l'édition frontend pour certains utilisateurs et groupes d'utilisateurs. Après s'être identifié, l'utilisateur se retrouve directement dans le frontend.

Si le rédacteur est identifié dans le backend, toutefois, il a la possibilité d'appeler des pages dans le mode d'édition frontend à l'aide du menu contextuel ou du bouton approprié. En règle générale, de petites icônes d'édition en forme de crayon [1] apparaissent sur tous les éléments de contenu et des barres d'édition [2]. Les visiteurs habituels du site ne voient bien sûr jamais ces éléments.

Figure 3.46:
Édition frontend à
l'aide des icônes [1]
et de la barre
d'édition [2]



Avec l'icône « crayon », les éléments de contenu sont affichés individuellement en mode d'édition dans une nouvelle fenêtre. La barre d'édition permet d'autres actions telles que déplacer, cacher et effacer le contenu, ou alors créer de nouvelles pages ou éléments de contenu.

Il est indispensable que l'administrateur ait configuré correctement les droits de lecture et d'inscription afin de pouvoir utiliser l'édition frontend. Il doit avoir activé le **Panneau d'administration** (Admin Panel) dans la configuration du gabarit et de l'utilisateur ou du groupe d'utilisateurs. À l'aide du Panneau d'administration, l'utilisateur peut, dans la section **Éditer**, choisir d'afficher ou non les icônes d'édition dans le frontend, ou encore d'afficher un champ de formulaire sur la page.

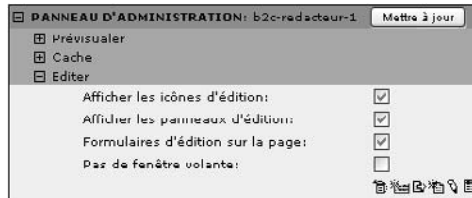


Figure 3.47:
Panneau
d'administration en
bas de page du
frontend

Les éléments dans l'édition frontend sont créés et affichés à l'aide des icônes de la barre d'édition. Le cache n'est pas utilisé dans ce mode d'édition. Même si les pages sont servies plus lentement à l'utilisateur, cela n'influence en rien la navigation d'un visiteur normal.

Le mode d'édition frontend fonctionne aussi longtemps que l'utilisateur est identifié dans le backend. Si la session backend prend fin (ou si le rédacteur se déconnecte), les icônes d'édition disparaissent.

3.9 Le Rich Text Editor

Pour les éléments de contenu **Texte** et **Texte & image**, il est possible d'utiliser le *Rich Text Editor* (RTE). Suivant la configuration, le rédacteur peut rédiger des blocs de texte en mode WYSIWYG (« what you see is what you get »)⁷ et, par exemple, insérer des images en les faisant glisser jusqu'à l'endroit souhaité, ou établir des liens. Toutefois, lorsque votre élément de contenu est sauvegardé, contenu et mise en forme ne font plus qu'un, étant donné que les balises HTML sont enregistrées directement dans le texte. C'est pourquoi, l'accès au RTE et à ses fonctions doit être soigneusement paramétré. Si une application doit par exemple respecter les règles d'accessibilité définies par les normes du W3C, avec des feuilles de style (Stylesheets) en cascade, le RTE ne doit bien sûr utiliser aucune balise pour les polices. La configuration sera abordée au chapitre 4.8.4.

Référence 888606

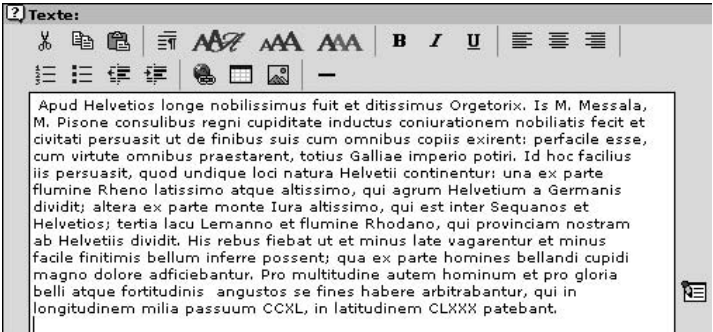
Le RTE dans TYPO3 s'appuie sur ActiveX, ce qui implique qu'il ne peut être utilisé qu'avec Internet Explorer sur des plate-formes Microsoft. Vous avez aussi la possibilité d'utiliser `htmlArea`, un autre logiciel libre. Vous trouverez la documentation relative à ActiveX en vous reportant à la référence ci-contre. Si le rédacteur a l'autorisation d'utiliser RTE, il peut le paramétrer (dans le module **Utilisateur** → **Configuration** ou en cochant les cases du formulaire d'édition situées en bas de page). Si le RTE est activé, une barre d'édition comportant plusieurs options apparaît dans la section **Texte** du formulaire.

Référence 788773

En cliquant sur l'icône à droite du champ texte, vous ouvrez le RTE en mode plein écran. Les changements apportés à l'original peuvent être visualisés en cochant la case **Code source**. Le RTE peut aussi être désactivé pour des éléments de contenu isolés en cochant **Désactiver Rich Text Editor** dans la section **Texte** de leur formulaire de création.

⁷NdT : mot à mot « ce que vous voyez est ce que vous obtiendrez »

Figure 3.48:
Le Rich Text Editor et
tous ses menus



Le rôle de la plupart des options d'édition est évident. De plus, en positionnant la souris sur les icônes, une courte description apparaît. Nous vous en donnons une vue d'ensemble dans le tableau suivant. Nous reprendrons plus loin les fonctionnalités plus complexes.

Tableau 3.2:
Aperçu des fonctions
du RTE

Fonction	Raccourci	Description
Couper	(Ctrl)+(X)	Coupe un élément de texte et le conserve dans la mémoire tampon (aussi disponible dans le menu contextuel).
Copier	(Ctrl)+(C)	Le texte sélectionné est copié dans la mémoire tampon (aussi disponible dans le menu contextuel).
Coller	(Ctrl)+(V)	Le texte contenu dans la mémoire tampon est inséré à l'endroit où est placé le curseur (aussi disponible dans le menu contextuel).
Style de paragraphe		Permet de sélectionner le format du paragraphe mis en surbrillance.
Style de caractère		Des classes prédéfinies par l'administrateur sont disponibles afin de mettre en évidence des extraits de texte.
Police de caractère		Le texte en surbrillance est affiché dans la police standard sélectionnée. En fonction des polices installées sur le PC du visiteur, le texte sera affiché dans la police sélectionnée ou non.
Taille de caractère		La taille du texte en surbrillance est ajustée sur une échelle de 1 à 7, conformément au standard HTML.
Couleur de texte		En fonction de la configuration, une palette ou bien une liste de couleurs prédéfinies s'affiche.
Gras	(Ctrl)+(B)	Le texte en surbrillance est affiché en gras (aussi disponible dans le menu contextuel).

suite

Fonction	Raccourci	Description
Italique		Le texte en surbrillance est affiché en italique (aussi disponible dans le menu contextuel).
Souligné		Le texte en surbrillance apparaît souligné (aussi disponible dans le menu contextuel).
Justifié à gauche		Le bloc de texte dans lequel se trouve le curseur est justifié à gauche.
Centré		Le bloc de texte dans lequel se trouve le curseur est centré.
Justifié à droite		Le bloc de texte dans lequel se trouve le curseur est justifié à droite.
Liste numérotée		Les paragraphes mis en surbrillance sont organisés en liste, dont chaque élément est précédé d'un nombre.
Liste		Les paragraphes mis en surbrillance sont organisés en liste, dont chaque élément est précédé d'une icône.
Diminuer l'indentation		Réduit ou retire l'indentation du paragraphe mis en surbrillance.
Augmenter l'indentation		Indente le paragraphe mis en surbrillance une ou plusieurs fois.
Insérer un lien		Les passages de texte mis en surbrillance deviennent des liens vers d'autres pages de l'application.
Insérer un tableau		Ouvre un assistant à la création d'un tableau.
Couleur d'arrière-fond		Comme pour la couleur du texte, une palette ou une liste de couleurs prédéfinies s'affiche en fonction de la configuration du RTE.
Insérer une image		Même si TYPO3 prévoit un type de contenu permettant l'insertion d'images dans un texte, cette opération peut être effectuée directement dans le RTE.
Insérer une icône		TYPO3 contient une liste d'icônes pouvant être insérées dans le texte. Vous pouvez ajouter de nouvelles icônes à cette sélection.
Insérer une ligne		Introduit une ligne entre les paragraphes.
Insérer des éléments utilisateur		Si l'administrateur a défini des fonctionnalités et des détails particuliers, ils seront introduits dans le menu approprié.

Insérer un lien

Référence 648217

Si un extrait de texte est mis en surbrillance et que vous appelez ensuite la fonction **Insérer un lien** dans la barre d'édition, le navigateur d'éléments s'ouvre avec des options spéciales pour l'édition. Comme d'habitude, vous pouvez « parcourir » la page ou l'arborescence de fichiers afin de définir un lien vers des pages, des éléments de contenu ou des fichiers contenus dans le système. Vous pouvez spécifier dans le menu de sélection, ou manuellement dans le champ **Cible** si le lien doit être ouvert dans sa propre fenêtre (**_self**), dans le haut du cadre (**_top**) ou dans une nouvelle fenêtre (**_blank**). Lorsque vous renvoyez à des **URL externes**, assurez-vous que vous inscrivez l'adresse complète, avec le protocole (**http://**). En ce qui concerne les **emails**, l'adresse du destinataire doit être insérée.

Figure 3.49:
Insertion de liens
avec le Rich Text
Editor



Insérer une image

Référence 033547

Les images peuvent aussi être insérées directement dans le texte à l'aide du RTE. Si la fonction **Insérer une image** est appelée dans la barre d'édition, le navigateur d'éléments affichera trois modes : **Nouvelle « Magic » image**, **Nouvelle « plain » image** et **Glisser-déposer**. Le mode **Nouvelle « Magic » image** supporte tous les formats et toutes les tailles d'image. L'image est automatiquement convertie vers un format graphique adapté au Web. Vous aurez plus tard la possibilité d'en modifier la taille afin d'optimiser le format et la qualité d'affichage. Le mode **Nouvelle « plain » image** n'accepte que les formats graphiques Web (JPG, GIF, et PNG), avec une résolution maximale de 640×480 pixels. Ce mode est prévu pour travailler avec des images déjà conçues pour le Web. Dans le mode **Glisser-déposer**, les images peuvent être insérées directement dans le texte à partir du navigateur d'éléments. Si une image déjà insérée a besoin d'être rééditée, vous devez la sélectionner et rappeler la fonction **Insérer une image**. Les attributs tels que la hauteur, la largeur, la bordure, la marge à gauche/à droite, la marge en haut/en bas ou le titre, peuvent être spécifiés dans le mode **Image courante**. Il est toutefois recommandé d'utiliser les éléments de contenu **Image** et **Texte & image** pour l'insertion d'images dans une page.

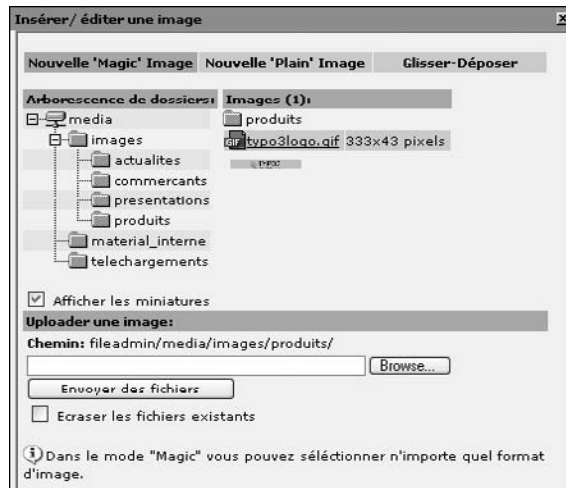


Figure 3.50:
Insertion d'images
dans le Rich Text
Editor à l'aide du
navigateur
d'éléments

Insérer un tableau

Un assistant simple d'utilisation permet d'insérer des tableaux dans le RTE. Son fonctionnement est plutôt rudimentaire : il permet de spécifier le nombre de lignes et de colonnes, ainsi que l'espacement inter- et intra- cellules, la couleur et la largeur des bordures, ou même une image d'arrière-fond pour tout le tableau. Toutefois, pour insérer un tableau, il est recommandé d'utiliser l'élément de contenu prévu à cet effet plutôt que le RTE.

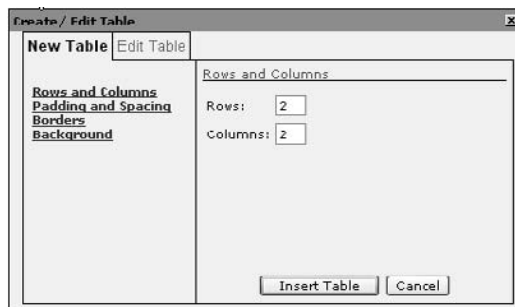


Figure 3.51:
Assistant à la
création d'un tableau
dans le Rich Text
Editor

3.10 Travailler efficacement avec TYPO3

Jusqu'ici, nous vous avons présenté l'édition dans le backend et dans le frontend, ainsi que les éléments de contenu standards. Dans cette section, nous abordons à l'aide d'un exemple pratique la manière de travailler efficacement avec TYPO3. Nous vous présentons les fonctions et les méthodes de travail qui simplifieront et accéléreront votre travail au quotidien.

3.10.1 Scénario

Notre exemple est basé sur un scénario de gestion de contenu dans lequel nous devons fournir de l'information à trois groupes cibles différents d'une installation TYPO3 :

1. **Business to Consumer (B2C)**
Site Web public destiné aux consommateurs ; cette partie de l'arborescence contiendra de l'information prévue pour un large public. Notre société fictive désire une page réservée à sa gamme de produits, une section d'actualités sur la page d'accueil, une page « événements », la liste de ses revendeurs et des éléments de contenu standards tels qu'un menu contact, le logo de la société, et un plan du site, repris dans la section Fonctions du site.
2. **Business to Business (B2B)**
Section réservée aux partenaires et fournisseurs et protégée par un mot de passe ; cette page contient de l'information plus détaillée sur les produits ainsi que du matériel de promotion pour les revendeurs.
3. **Business to Employee (B2E)**
Intranet réservé aux employés ; ce forum destiné à l'usage interne contient de l'information pour les employés.

Un tel scénario est idéal pour la réutilisation du même contenu à différents endroits. Malheureusement, il en est rarement ainsi dans la pratique. Pour des raisons de sécurité, les serveurs intranet et Internet sont toujours séparés, de sorte que la situation décrite plus haut n'est possible que si tous les sites Web sont créés sur le serveur intranet et que les sites B2B et B2C sont alimentés en contenu par un autre serveur Web situé sur Internet. Le chapitre 2.6 donne plus d'informations sur une architecture pouvant servir de solution, en particulier sur la séparation entre le serveur en ligne et le serveur de production.

3.10.2 Créer l'arborescence des pages

Référence 189036

Nous commençons d'abord par créer les trois sites Web dans le système.

Figure 3.52:
Arborescence du site



Nous créons d'abord un « Portail », et une page « B2C Accueil » au niveau inférieur. Une méthode plus rapide est appliquée pour créer les sous-pages. Dans le module **Web** → **Fonctions**, un **Assistant** présente un formulaire permettant de créer jusqu'à neuf pages simultanément. L'assistant place ces pages après la page sélectionnée dans l'arborescence, ou en tant que sous-pages de celle-ci. Dans notre exemple, vous commencez par créer la page « B2C Accueil » manuellement, puis vous l'appellez dans le module **Web** → **Fonctions**. Vous

sélectionnez ensuite l'assistant dans le menu déroulant dans le coin supérieur droit de la vue détaillée, et la fonction **Créer plusieurs pages multiples**. Saisissez les titres des pages que vous souhaitez créer. Vous ne remplissez pas les champs dont vous n'avez pas besoin. Les nouvelles pages seront créées en tant que sous-pages, à moins que vous ne spécifiez le contraire en cochant la case **Placer les nouvelles pages à la suite**. L'option **Masquer les nouvelles pages** indique que les pages nouvellement créées ne seront pas visibles dans le frontend, ce qui devrait toujours être le cas lorsqu'on ajoute des pages aux sites Web déjà existants.

La fonction **Trier les pages** vous permet de trier les pages d'un même niveau en fonction du titre, du sous-titre, de la date de modification ou de la date de création, ou encore d'en inverser l'ordre actuel.



Figure 3.53:
Assistant du module
Web → Fonctions
pour faciliter la
création et le tri de
pages

Afin de créer l'arborescence B2B, configurez l'option **Copie récursive** du module **Utilisateur** → **Configuration** afin de copier au moins un niveau lorsque vous enregistrez une page dans la mémoire tampon. Vous pouvez ensuite insérer la page copiée après la page « B2C Accueil » en cliquant sur l'icône d'« B2C Accueil » et en y ajoutant la page copiée avec l'option **Coller après**.

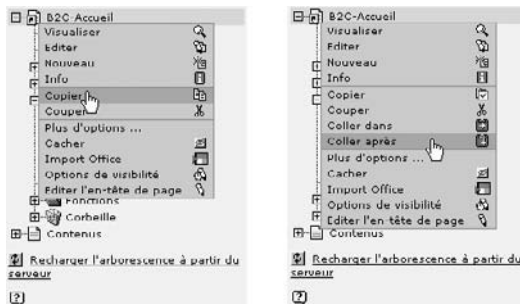


Figure 3.54:
Copier-coller d'une
page

3.10.3 Presse-papiers

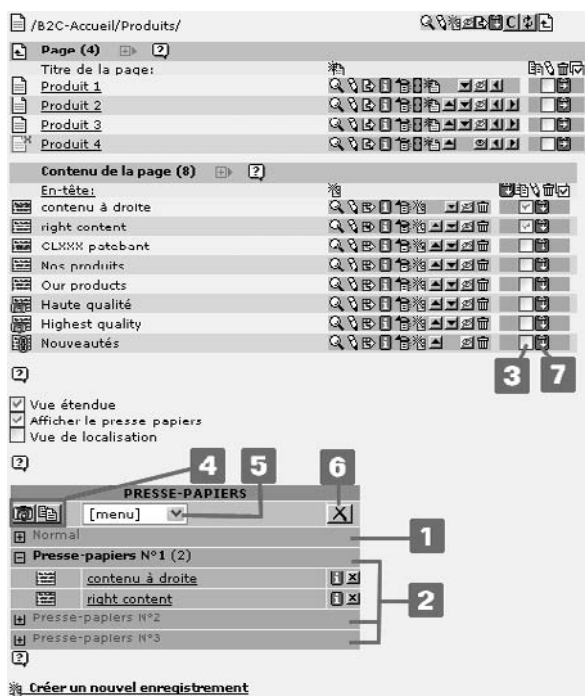
Référence 965454

Il est souvent nécessaire d'éditer plusieurs pages ou éléments de contenu au cours d'une session de travail ; on utilise alors le **Presse-papiers**, qui affiche les éléments que contient la mémoire tampon et propose une série d'options d'édition.

On l'active en cochant la case **Afficher le presse-papiers** dans le bas de la vue détaillée. Vous devez aussi cocher la case **Vue étendue** si elle n'est pas déjà activée. Alors que la mémoire tampon **Normal** [1] ne peut contenir qu'un élément à la fois, les **Presse-papiers 1-3** [2] peuvent contenir plusieurs pages ou éléments de contenu.

Si le presse-papiers est activé, une icône supplémentaire apparaît dans la vue détaillée du mode **Liste** (repris ci-après sous le nom « affichage liste »). Dès qu'un des **Presse-papiers 1-3** est activé, des cases à cocher apparaissent derrière chaque élément listé, rendant possible la sélection de plusieurs éléments [3]. L'icône « information » dans la barre de menu (cf. vue étendue) affiche toute l'information essentielle à propos de l'enregistrement en question.

Figure 3.55:
Utilisation du
presse-papiers : le
presse-papier n° 1 est
actif et contient deux
éléments



Dans le presse-papiers, l'icône « appareil photo » permet d'afficher les vignettes des images copiées dans le presse-papiers, et l'icône « copier » permet de choisir entre le mode « copier » (icône orange) ou « déplacer » (icône grise) pour tous les éléments contenus dans le presse-papiers [4]. L'endroit où les éléments doivent être insérés est déterminée à l'aide de l'icône correspondante dans la vue détaillée (**Coller dans ; Coller après**) [7]. Mais il est aussi possible d'ouvrir les enregistrements dans le presse-papiers pour les éditer ou les supprimer définitivement à l'aide du menu déroulant [5]. Si plusieurs éléments se trouvent dans

la mémoire tampon, ils seront affichés en tant que formulaires d'édition, l'un à la suite de l'autre. Si vous souhaitez simplement effacer les enregistrements de la mémoire tampon, cliquez sur l'icône marquée d'une croix à droite de chaque élément, ou sur le bouton dans le menu d'édition du presse-papiers [6].

Exemple

Dans notre exemple, la première copie de l'arborescence de B2C est renommée en « B2B Accueil » et on y ajoute deux nouvelles pages : « Téléchargements » (après « Revendeurs ») et un dossier système, « FEUtilisateur » (après « Fonctions »). Une fois le presse-papiers activé, sélectionnez toutes les sous-pages de « B2B Accueil » dans l'« affichage liste » et choisissez l'option **Éditer la sélection**.



Figure 3.56:
Options de sélection
et d'édition dans
l'« affichage liste »
quand le
presse-papiers est
activé

Dans les champs Titre dans la liste de formulaires, vous pouvez maintenant modifier les pages afin d'obtenir l'arborescence suivante.



Figure 3.57:
Arborescence de la
section « B2B
Accueil » après
édition et ajout de
deux nouvelles pages

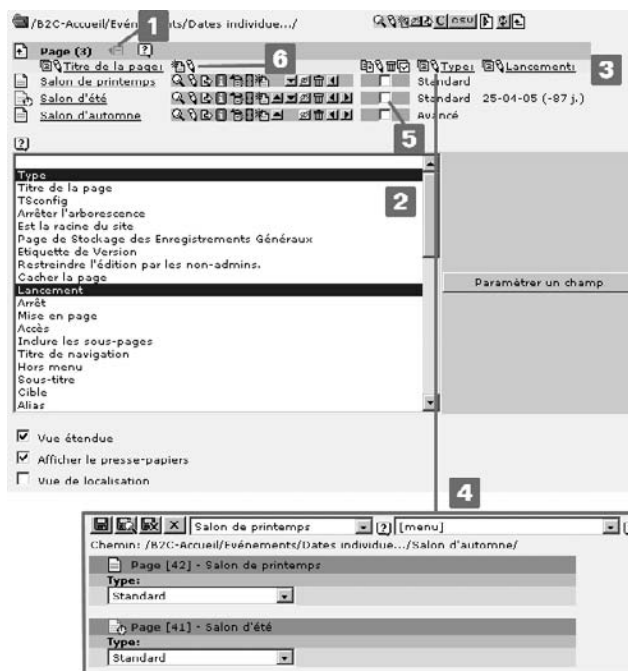
À partir de la page « Commerçants », une page « Revendeurs » a maintenant été créée. L'arborescence B2E doit encore être créée ; nous procéderons en utilisant les options de traitement par lots dans l'affichage liste.

3.10.4 Éditer des champs sélectionnés

L'affichage liste offre la meilleure vue d'ensemble, ainsi que le meilleur accès aux enregistrements liés à une page. Mais elle offre d'autres fonctionnalités qui ne sont pas visibles au premier coup d'œil. Nous vous recommandons de commencer par activer la **Vue étendue** et le **Presse-papiers**. Pour tous les types de données (tels que les pages, les éléments de contenu, etc.), on peut sélectionner les tables correspondantes, afin de les visualiser ou de les paramétrer individuellement, à l'aide de l'icône « + » située dans l'en-tête [1].

Référence 729118

Figure 3.58:
Dans le module Liste,
les champs Type
correspondants sont
ouverts pour leur
édition via la table
Page et la sélection
de deux
enregistrements



En conséquence, l'« affichage liste » est élargi grâce à un menu de sélection où les champs individuels peuvent être sélectionnés, afin d'être ensuite affichés dans la table [2]. Dans cette figure, on a sélectionné les champs **Type** et **Lancement** [3]. En cliquant sur le titre des champs, vous pouvez trier les enregistrements de cette sélection par ordre croissant ou décroissant. Vous pouvez ainsi trier rapidement les nouveaux éléments par ordre alphabétique, ou par date de création, par exemple. Tous les champs affichés dans une colonne peuvent être édités grâce à l'icône « crayon » [4]. En combinant cette fonction avec un presse-papiers et en marquant les enregistrements individuellement, vous pouvez restreindre la sélection de données pour leur édition. Afin d'ouvrir le formulaire d'édition pour tous les champs des colonnes et des enregistrements listés (ou sélectionnés), cliquez sur l'icône « crayon » située au-dessus des options d'édition individuelle [6].

Cette méthode facilite grandement l'édition de la méta-information (résumés, mots-clés et descriptions) dans l'arborescence pour toutes les pages déjà créées à un même niveau. L'affichage des champs est aussi conservé et restitué dès que l'utilisateur clique sur l'icône « Menu » (le symbole « - ») afin d'afficher toutes les tables. Toutefois, elles ne peuvent être éditées individuellement dans ce mode.

Une seconde option consiste à exporter des éléments comme un fichier CSV, qui apparaît si vous avez sélectionné une table en affichage exclusif dans le module **Liste**. Elle permet de télécharger le contenu des champs sélectionnés plus tôt en tant que fichier CSV (*Comma Separated Values*)⁸

⁸NdT : valeurs séparées par des virgules.

Exemple

Afin de compléter notre arborescence, nous copions maintenant l'arborescence « B2B Accueil » et insérons la copie après l'original. Cette nouvelle page est renommée « B2E Accueil ». Les fonctions additionnelles sont maintenant visibles dans l'« affichage liste » grâce à l'icône « + » dans l'en-tête de la table. Sélectionnez **Type**, **Titre de la page** et **Raccourci vers page**. Si vous cliquez sur l'icône « crayon » adjacente au titre de colonne correspondant, tous les éléments de cette colonne seront affichés dans un formulaire d'édition.

Maintenant renommez les pages, modifiez les types de pages et corrigez le raccourci de la page « Accueil » de façon à produire l'arborescence suivante.

Titre de la page:	Type:	Raccourci vers page:
Accueil	Raccourci	75
Nouveautés	Avancé	
Événements	Avancé	
Départements	Avancé	
Fournitures Bureau	Avancé	
Adresses	Avancé	
Manuels	Standard	
Panneau Affichage	Avancé	
Fonctions	Dossier Système	
Corbeille	Corbeille	
FEUtilisateurs	Dossier Système	

Figure 3.59:
Liste en mode étendu
après édition

3.10.5 Raccourcis

Après avoir travaillé un certain temps avec TYPO3, vous réaliserez que vous devez éditer certaines pages — par exemple les pages d'actualités — particulièrement souvent. Plusieurs raccourcis sont disponibles afin de vous permettre d'accéder directement à ces pages après vous être identifié.

Utilisateur → Centre de tâches : Pages récentes

Si la page souhaitée a été éditée récemment, elle peut être ouverte directement dans le module **Utilisateur → Centre de tâches** sous la rubrique **Pages récentes**, après que vous vous êtes identifié.

Utilisateur → Centre de Tâches :

L'administrateur peut ajouter à la liste les pages ayant besoin d'être souvent éditées, et y donner un droit d'accès dans le module **Utilisateur → Centre de tâches**.

Raccourcis en bas de page

Une autre possibilité est de créer vos propres raccourcis vers les pages correspondantes. L'administrateur a déjà cette possibilité dans la configuration de base du profil utilisateur. Pour les

Référence 585979

rédacteurs, l'administrateur doit accorder les droits via TSConfig, afin que la barre de raccourcis soit affichée au bas de la page. Voir la référence ci-contre.

```
options.shortcutFrame=1
```

Le rédacteur a maintenant la permission de créer des liens vers la page voulue via l'icône de raccourci [1] dans la vue détaillée. Ils apparaissent en bas de page du backend [2] et peuvent être organisés en groupes [3]. Pour y parvenir, activez l'option **Éditer**, cliquez sur le lien, assignez-le à un groupe et nommez-le. Le titre permet d'accéder au lien directement via le menu de sélection [5].

Référence 348238

De plus, le rédacteur a accès à un champ [6] qui permet d'appeler les pages de l'arborescence directement en entrant leur ID, par exemple, si des erreurs d'affichage ont été constatées dans le frontend.

Figure 3.60:
Barre de raccourcis
dans le bas de page
du backend



3.10.6 Aide au niveau du contenu

Recherche et remplacement de texte avec l'extension **Text tools**

Même si les pages et la quantité de données grandissent rapidement au sein d'une application, TYPO3 peut vous aider à conserver une vue d'ensemble : les enregistrements et éléments de contenu spécifiques peuvent être retrouvés facilement à l'aide du champ de recherche situé au bas de la vue détaillée. La recherche peut s'étendre jusqu'à trois niveaux dans l'arborescence.

Figure 3.61:
Options de recherche
dans la vue détaillée



Référence 325093

Une façon encore plus efficace de trouver des éléments de texte et de les éditer ou de les remplacer immédiatement est d'utiliser l'extension **Text tools** (`cc_textfunc`) du module **Web** → **Fonctions**. L'extension doit d'abord être installée comme décrit à la section 5.7 pour apparaître comme fonction dans le module **Web** → **Fonctions**. Vous pouvez maintenant parcourir des tables, des pages, des éléments de contenu ou des gabarits jusqu'à quatre niveaux. Via la fonction **Search and replace**, tous les enregistrements trouvés sont listés avec leur contenu existant et modifié. De plus, vous pouvez aussi spécifier quels sont les enregistrements où les modifications seront apportées.

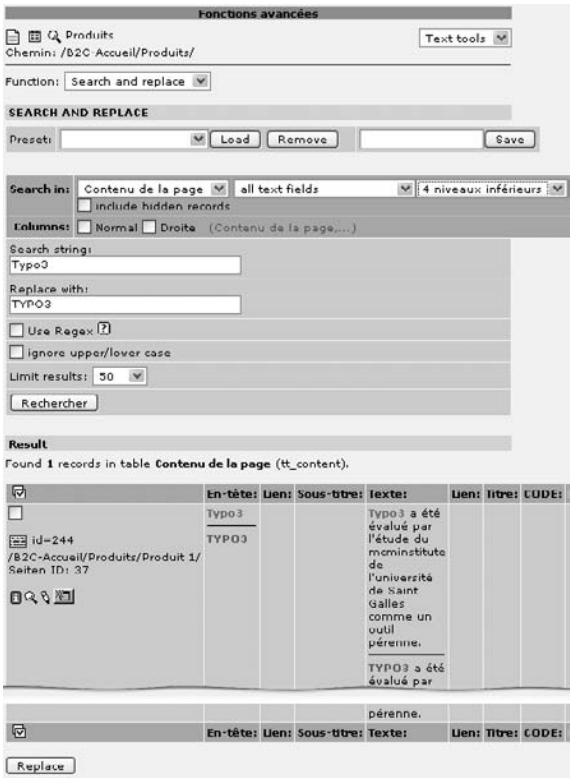


Figure 3.62:
L'extension Text Tools
permet facilement
l'opération
rechercher/remplacer

Intégrer des documents Office

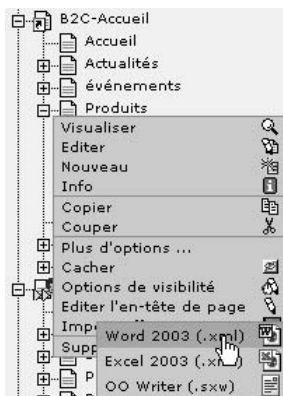
Lorsqu'il génère du contenu, le rédacteur est souvent confronté à des fichiers basés sur des documents de la suite Office. Si des sections du document doivent être utilisées en tant qu'élément de contenu, il est nécessaire de décider si les détails de mise en forme doivent être repris ou non dans TYPO3. En théorie, les détails de mise en forme peuvent être repris en l'état par le Rich Text Editor, en fonction du processus de transformation configuré par le développeur. Si vous souhaitez ne pas tenir compte de la mise en forme des documents Office et spécifier vos propres balises, il est conseillé d'enregistrer les textes dans un éditeur ASCII, ou de désactiver le Rich Text Editor avant d'insérer le texte. On peut ensuite réactiver le RTE pour éditer le texte.

Si vous voulez inclure non seulement des extraits de texte, mais aussi des documents Office dans leur ensemble, nous recommandons alors d'utiliser l'extension **General Office Displayer** (bientôt renommée **Document Suite**). Cette extension propose une intégration transparente de documents Word 2003, Excel 2003 et OpenOffice Writer. Alors que les deux premiers formats doivent être enregistrés explicitement en tant que documents XML, le format SXW est supporté pour les documents OpenOffice. Deux méthodes d'intégration sont maintenant disponibles pour le rédacteur.

Référence 138944

Dans le premier cas, le document est importé vers le Rich Text Editor : sélectionnez la page de l'arborescence où le contenu doit être inséré, activez le menu contextuel et cliquez sur le menu **Import Office** en choisissant le format approprié. Le contenu est alors enregistré en tant que type de contenu **Texte**, et peut ensuite être édité par le RTE.

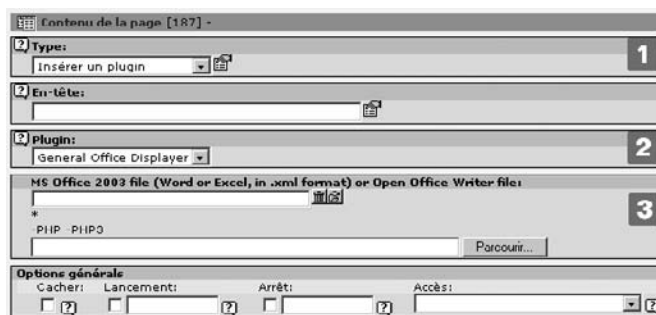
Figure 3.63:
Import Office via le
menu contextuel



La seconde méthode intègre le document souhaité à l'aide d'un plugin. Au cours de ce processus, le document est extrait de son propre format et transformé en un élément de contenu sous une mise en forme prédéfinie. Le texte ne peut toutefois plus être édité. Le document est intégré avec un élément de contenu de type **Insérer un plugin** [1] à l'aide de l'extension **General Office Displayer** [2]. Le fichier Office peut maintenant être ajouté via le système de fichiers ou à partir du réseau local [3].

Le **General Office Displayer**, publié par Robert Lemke, continue à évoluer sous le nom **Document Suite**. La gamme de fonctionnalités sera considérablement élargie, et un certain nombre d'options de manipulation seront ajoutées.

Figure 3.64:
Import d'un
document Office via
un plugin



3.10.7 Restaurer/éditer l'historique

Référence 763322

Un historique (simple) du contenu a été introduit dans la version 3.3 de TYPO3. Par défaut, le contenu de toutes les tables est enregistré pendant sept jours (604 800 secondes), avec un maximum de dix versions pour chaque enregistrement.

L'administrateur peut paramétrer ce processus plus précisément pour les tables prises séparément via TCEMAIN_tables.

Référence 729572

Le rédacteur peut retourner à la dernière version en un seul clic. Si vous placez la souris sur l'icône « annuler » ou « rétablir », la date de la dernière modification apparaît.



Figure 3.65:
Bouton « rétablir » à droite de la barre d'édition

Le rédacteur peut visualiser l'historique des changements dans les modules **Web** → **Page** → **Édition rapide** et **Web** → **Liste** en cliquant sur l'icône prévue à cet effet.



Figure 3.66:
Icône pour l'historique des changements dans le module Liste

L'historique donne un aperçu des modifications apportées à l'enregistrement en question. Tous les changements sont indiqués en vert ; les valeurs existant depuis longtemps ou celles ayant été effacées sont indiquées en rouge.

CHANGEMENTS À L'ENREGISTREMENT		
Nom du champ:	Différence:	
Date et heure:	24-07-05 21:26, 7 min.	Utilisateur: b2c-redacteur-1
Texte:	Typo3 TYPO3 a été évalué, dans le cadre d'une étude confiée au mcminstitute de l'université de Saint Gallen, comme une solution pérenne.	
Date et heure:	24-07-05 21:28, 5 min.	Utilisateur: b2c-redacteur-1
Texte:	TYPO3 a été évalué, dans le cadre d'une étude confiée au mcminstitute de l'université de Saint Gallen, comme une solution pérenne.	
Désactiver Rich Text Editor:	Yes	
Date et heure:	24-07-05 21:29, 4 min.	Utilisateur: b2c-redacteur-1
Désactiver Rich Text Editor:	Yes	
Date et heure:	24-07-05 21:40, 4 min.	Utilisateur: b2c-redacteur-1
Texte:	 TYPO3 a été évalué, dans le cadre d'une étude confiée aumcminstitute de l'université de Saint Gallen, comme une solution pérenne.	
① Dans les différentes colonnes les textes en vert représente les nouveau changement et les textes en rouge les anciennes valeurs supprimées		
②		
RETOURNER		
🔗 Cliquez ici pour retourner en arrière		

Figure 3.67:
Affichage des historiques des changements

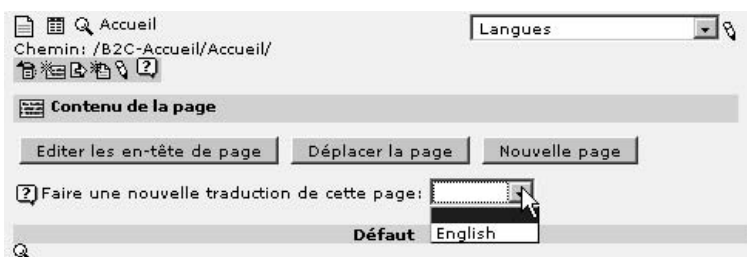
Tous les changements jusqu'à la version actuelle sont restaurés avec l'icône « crayon » alors qu'avec l'icône « information », une version de tous les changements jusqu'à la version actuelle est affichée.

3.10.8 Multilinguisme

Il est possible de travailler avec plusieurs langues dans une seule et même arborescence avec TYPO3. Si les gabarits de page sont préparés en conséquence, plusieurs versions d'une page pourront être éditées en fonction de la langue sélectionnée.

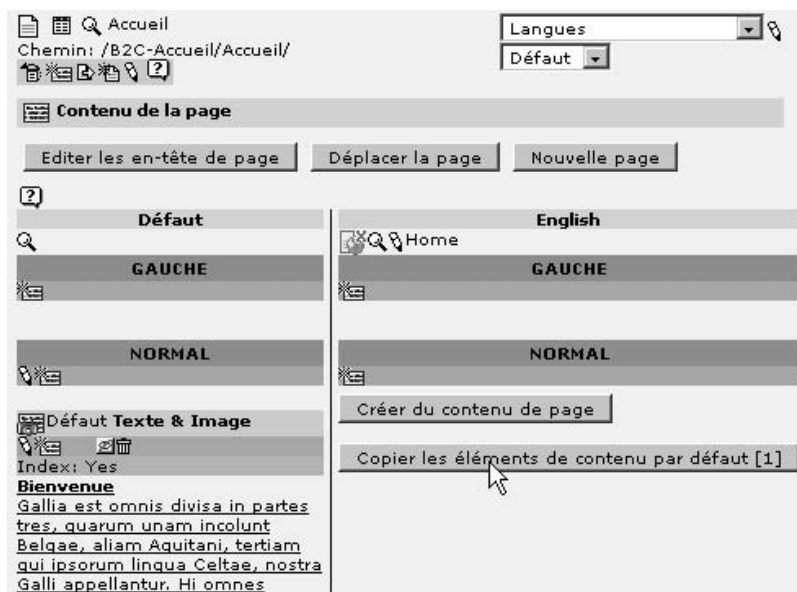
Une option de traduction apparaît alors à l'intention du rédacteur dans le module **Web** → **Page** sous l'option **Langues**.

Figure 3.68:
Création d'une
traduction



Après avoir sélectionné la langue pour la page en question, vous devez remplir les champs de la section **Titre de la page**. Vous pouvez alors **Créer du contenu de page** ou **Copier les éléments de contenu par défaut**.

Figure 3.69:
Affichage du contenu
dans chaque langue



Cette dernière possibilité offre l'avantage de pouvoir envoyer au rédacteur les éléments à traduire, selon les permissions qu'il a dans les différentes langues. La mention « Translate to *langue* » apparaît alors au début du texte à traduire.

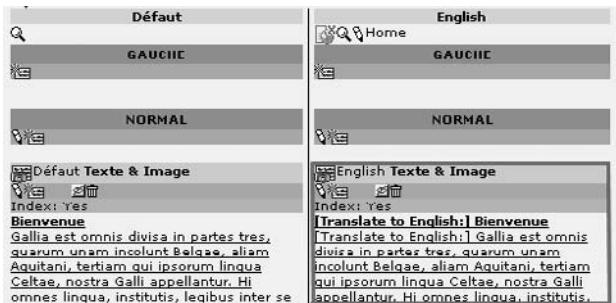


Figure 3.70:
Élément de contenu à traduire

Si vous ouvrez le formulaire d'édition de la traduction de la page, vous constaterez que les valeurs des champs sont reprises [1] et comparées à celles de la version originale (via les zones vertes) [2].

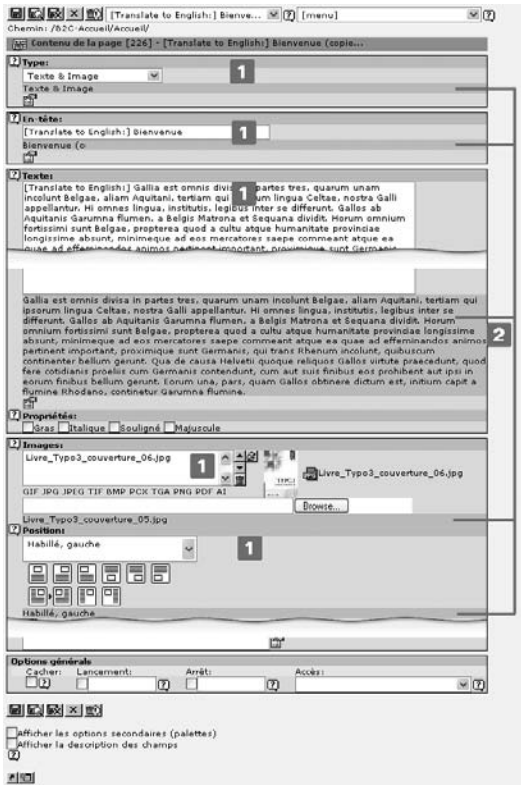
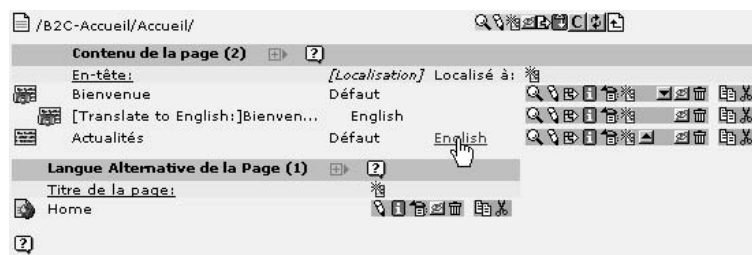


Figure 3.71:
Vue d'ensemble de l'enregistrement

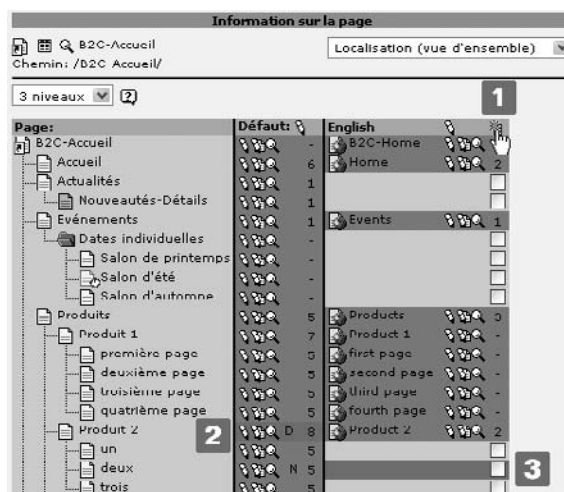
Vous pouvez aussi démarrer ce workflow de traduction directement dans le module **Web** → **Liste** ou activer la **Vue de localisation**. Pour les traductions manquantes, un message apparaît.

Figure 3.72:
Traductions
manquantes ou
incomplètes



Le module **Web** → **Info** → **Localisation** donne la meilleure vue d'ensemble sur des sections entières de l'arborescence ainsi que sur l'état des traductions. On ajoute des titres aux traductions des pages en cochant les cases des pages respectives, puis en cliquant sur l'icône **Créer un nouvel en-tête d'enregistrement** [1]. Il est possible d'éditer les en-têtes de pages ainsi que le contenu. Les autres options relatives à la traduction telles que **Cacher la traduction par défaut** (vert) [2] ou **Cacher la page si elle n'est pas traduite** (rouge) [3] sont mises en évidence par une couleur différente.

Figure 3.73:
Aperçu de l'état des
traductions



TYPO3 pour les administrateurs

4

Chapitre

TYPO3 pour les administrateurs

4.1 Tâches et objectifs de l'administration

Par administration, on entend toutes les tâches n'impliquant ni le travail d'installation et de programmation pour la mise en œuvre d'une part, ni la production de contenu d'autre part. L'administration comprend la maintenance technique du système, sa configuration, sa surveillance, ainsi que le contrôle et l'adaptation des résultats de la production de contenu. Jusqu'à présent, les occasions de s'exercer au rôle de gestionnaire de contenu ont été rares, et la plupart des personnes œuvrant dans cette branche n'ont pas cherché à y entrer, tout au contraire : le secteur les a trouvés. C'est toutefois une tâche critique pour l'entreprise, car il ne s'agit pas seulement de maintenir une bonne image de la société, mais aussi d'intégrer l'outil de gestion de contenu Web dans le champ plus large de la gestion. ¹

¹« People need to realize that the Web is no longer the thing about the thing ; it *is* the thing itself. The site represents your organization. Content management is, indeed, managing the business. » Suzanna Phelps-Fredette dans une transcription d'une conférence (« Content Management — How Can We Stop the Train Wreck? ») dans le cadre de la conférence IQPC Web Site Content Management tenue à San Francisco en 2000, <http://www.metatutorial.com/papers/aha.asp>

Vue de cette façon, « la gestion de contenu [...] s'intègre complètement dans la gestion d'information interne dans une société et dans la gestion des connaissances. Elle combine des aspects organisationnels, des procédés d'entreprise et des technologies. »²

Mais la technique n'est qu'un moyen : « alors qu'elle simplifie la création, l'enregistrement et la diffusion de contenu, fondamentalement, ce sont les procédés d'entreprise et les workflows qui permettent l'utilisation efficace et profitable de la technique. »³

L'administrateur joue le rôle d'un architecte présent en permanence, qui rend l'information visible à l'extérieur en organisant les ressources en conséquence. Que ce soit l'administrateur lui-même ou un consultant qui ait créé les processus de travail, l'administrateur doit être complètement familiarisé avec la terminologie et les méthodes du CMS utilisé, afin de pouvoir influencer la conception, l'évaluation et l'optimisation de l'information publiée.

Malheureusement, lors de l'introduction d'un CMS, on oublie souvent que l'effort principal réside dans la production continue de contenu. « Alors que la technique est nécessaire à l'implantation d'un CMS, cet investissement est la partie la plus simple d'une stratégie CMS. »⁴

Ce fait a été largement documenté et se vérifie dans pratiquement tous les projets CMS. De plus, l'introduction d'un CMS entraîne toujours des changements tels que de nouvelles tâches et de nouveaux procédés. Même lorsque le contenu est préexistant, des changements à large échelle sont probables : lors de l'introduction du CMS, les processus existants doivent aussi être améliorés, avec pour conséquences leur replanification et leur optimisation.

L'administrateur se trouve donc au cœur des opérations d'introduction et d'exécution ; il ou elle est responsable des processus concernés. Le travail qui doit être fait directement sur le CMS nécessite l'analyse des besoins pour le processus de rédaction, et leur confrontation avec les outils proposés dans le système.

Les gestionnaires de contenu ayant le rôle d'administrateurs contrôlant le système sont souvent des employés moins versés dans les technologies, et qui sont plutôt responsables de l'image externe de la société, du marketing et des relations publiques, ou de la communication avec les clients (B2C), les partenaires d'affaires (B2B), et les employés (B2E). Des assistants (wizards) les accompagnent dans la plupart de leurs tâches dans TYPO3. Ces utilitaires offrent une grande variété d'options de configuration pour l'interface utilisateur, et permettent également d'accorder certains droits aux groupes et aux utilisateurs. Même quand les paramètres du système sont impliqués, notamment avec le système TSConfig, un assistant guide ceux qui s'y connaissent moins bien en informatique.

L'administrateur a aussi la responsabilité de garantir la création de valeurs à partir des processus CMS. Dans ce cas, le système agit à la fois en tant qu'environnement de production et en tant qu'outil d'analyse. Dans ce domaine, TYPO3 offre plusieurs modules internes permettant l'analyse des statistiques. En plus des paramètres propres à la publication de contenu, il existe une puissante interface de base de données à l'aide de laquelle vous pouvez définir par vous-même des requêtes spécifiques, qui peuvent ensuite être enregistrées pour un usage ultérieur.

²Florian Stahl : « Dämme gegen die Informationsflut : Content Management ist mehr als ein Stück Software », Neue Zürcher Zeitung, 23.05.03, <http://www.nzz.ch/2003/05/23/em/page-article8TPZ8.html>.

³Même source

⁴Geoff Choo : « CMS strategy : Don't put the cart before the horse », TechRepublic Ins, 11 décembre 2001, ZDNet Australie : <http://www.zdnet.com.au/insight/toolkit/weboperations/cms/0,39023923,20262306,00.htm>

Dans ce cadre, le chapitre qui suit vous présente les problèmes principaux rencontrés lorsqu'on met en place un environnement de gestion de contenu, et illustre par le biais d'exemples la définition des droits, diverses options de configuration pour le backend, ainsi que des outils de contrôle et de validation du contenu.

4.2 Planifier et installer l'environnement de gestion de contenu

C'est une tâche ardue que d'analyser l'introduction d'une application de gestion de contenu à travers des processus nouveaux ou existants. Pareil travail implique la définition des sources, des fréquences et des formats de contenu. De plus, il faut aborder la planification et la mise en place des étapes de travail afin de répondre aux besoins rédactionnels du projet.

Le *Business Process Redesign* (BPR) est un outil répandu pour l'analyse, le développement et la construction de tels processus de travail et leur mise en œuvre dans un environnement de gestion de contenu. Le BPR combiné à un **Rapid Application Development** (RAD) produit des workflows efficaces et des processus d'entreprise supportés par une infrastructure informatique. Nous expliquerons brièvement ces deux approches. Nous vous donnerons par la même occasion une base à partir de laquelle vous apprendrez à mieux connaître ces procédures.

Dans ce cadre, il suffit de comprendre que, dans le contexte du BPR, chaque processus d'entreprise peut être disséqué de façon analytique en une chaîne d'éléments. Cette analyse met en évidence le déroulement d'une tâche spécifique, avec toutes les conditions, les transformations, les résultats et les décisions qui s'y rattachent. Cela permet d'identifier les potentiels d'optimisation, qui souvent peuvent être supportés de façon pratique par l'utilisation ou la modification de la technologie de CMS.⁵

Le choix d'un système de gestion de contenu résulte souvent d'une telle approche analytique, qui peut ici faire office d'exemple d'optimisation fondamentale du processus de gestion de contenu : sans CMS, la création et la maintenance de sites Web demeurent réservées aux experts ayant des connaissances techniques. Le chemin allant de la création d'information au sein de la société jusqu'à sa communication avec la clientèle, ses employés ou des partenaires d'entreprise est devenu beaucoup plus efficace et rapide grâce au CMS.

En effet, les employés ayant la connaissance du contenu nécessaire sont maintenant en mesure de la publier sans expertise technique : le processus de publication d'information s'en trouve simplifié et accéléré.

Comme second exemple, un CMS pourrait dupliquer automatiquement toute donnée produite par une source utilisable universellement, et afficher les annotations faites par les rédacteurs, puis les publier sur des sites Web, par le biais de pages statiques, ou via des services Web. La maintenance de données à l'état pur serait optimisée par la centralisation et la disponibilité de données enrichies en une sorte de noyau d'information multi-usage. Un exemple de ce genre peut être trouvé à la référence ci-contre.

Référence 253617

⁵Vous pouvez consulter une représentation théorique exhaustive accompagnée d'exemples pratiques dans « Workflow Management : Models, Methods and Systems » de Wil van der Aalst et Kess van Hee, The MIT Press, Cambridge Massachusetts, 2002.

Les procédures d'entreprise devraient être réalistes, faciles à utiliser et pragmatiques. Aussi souvent que possible, la création et le traitement de l'information doivent être groupés, les activités parallèles fusionnées, les mécanismes de contrôle mis en œuvre, et l'information recueillie à la source.⁶

Ces mots d'ordre peuvent, à bien des égards, être adoptés dans une situation de gestion de contenu, que ce soit par la centralisation des données à l'échelle de l'entreprise dans une source unique, ou par l'assignation intelligente des droits d'utilisateur, en configurant l'interface de rédaction en fonction des tâches à effectuer, par des procédures transparentes et simples, ou encore par des contrôles de qualité permanents.

Avec TYPO3, les procédés planifiés de cette façon peuvent être supportés et affichés concrètement par le système de droits d'accès, par la configuration optimisée de l'interface rédacteur, et par des workflows et des commandes prédéfinis. De plus, on peut développer et ajouter des extensions spécialisées afin d'améliorer ce processus.

Cette analyse méthodique et cette conception du processus conduisent à la description détaillée de la solution envisagée :

- Le site est structuré par sujets
- Le contenu, les sources et les formats (actualités, études de cas, produits, feuilles de données, etc.)
- Les utilisateurs sont regroupés par rôles
- Les processus de travail des utilisateurs sont en relation avec les éléments de contenu/formats ainsi qu'avec le flux des informations dans ce qu'on appelle le cycle de vie des éléments de contenu (cf. section 1.2.2)

4.3 Principes d'organisation des droits d'accès dans TYPO3

Alors que l'organisation de contenu peut être directement injectée dans la structure du site, l'analyse des droits à accorder à chaque utilisateur n'est pas menée de façon aussi intuitive ; il faut connaître le mode de fonctionnement d'attribution des droits dans le système. On distingue principalement les trois points suivants :

Utilisateurs et groupes

Des paramètres globaux peuvent leur être assignés afin de contrôler les options d'édition.

Pages

Pour chaque page, on définit les permissions pour le « propriétaire », le « groupe » et « tous » (tous les autres).

Contrôle de l'interface d'édition

L'interface utilisateur peut être paramétrée en fonction des pages ou en fonction des utilisateurs/groupe d'utilisateurs.

⁶M. Hammer : "Reengineering Work : Don't Automate, Obliterate", Harvard Business Review, July-August 1990, pp.104-112

Dans la pratique, la structure du contenu existant détermine la structure future des pages dans une arborescence. Les droits des employés concernés sont liés à cette structure de pages en fonction des processus de travail.

En se basant sur les types de tâches, on peut mener une analyse des fonctionnalités nécessaires à chaque profil d'utilisateur et envisager les modifications à apporter à l'interface pour simplifier son utilisation, ainsi que pour diminuer les coûts de formation.

Le résultat peut être présenté formellement dans un diagramme à deux dimensions dont les axes sont représentés respectivement par les utilisateurs et les droits. Très souvent, il suffit de convertir directement ce diagramme dans les paramètres correspondants de TYPO3. Le module **Outils** → **Administration des utilisateurs** vous donne une vue d'ensemble sur les droits des utilisateurs. Cette manière de procéder est logique et rapide. Elle convient particulièrement bien au contexte de l'approche RAD décrite plus bas.

Cette approche est basée sur le principe selon lequel le succès du projet dépend du degré d'implication des utilisateurs dans le processus de définition, et de leur niveau de connaissance. Lors du développement de logiciels, la méthode RAD (*Rapid Application Development*) a été développée à cette fin, ce qui implique la validation de l'approche conceptuelle d'un projet informatique en développant des prototypes de l'application dès les tout premiers stades. Par la suite, les utilisateurs sont aussi sollicités pour tester et améliorer l'application.⁷

Cette approche garantit que la planification et la mise en œuvre demeurent étroitement liées, et qu'il n'y aura aucun choc désagréable lors de la mise en application de la théorie dans des situations quotidiennes.

TYPO3 est idéal pour la création rapide de prototypes, afin par exemple de tester en pratique la distribution des tâches et des droits chez les utilisateurs. Un autre point positif est que les utilisateurs se familiarisent déjà avec les options et les principes fonctionnels, se préparant de la sorte au travail qui les attend.

Grâce à la possibilité qu'offre TYPO3 d'ajouter du contenu et de créer des structures de pages, même sans que l'interface soit complètement finie, vous pouvez commencer à vous entraîner immédiatement après l'installation et la configuration. De cette façon, vous comprendrez plus rapidement les concepts d'utilisateur et de structure du contenu, et la connaissance que vous en aurez pourra être utilisée de façon appropriée afin de concevoir l'interface, et de produire des applications.

4.3.1 Exemple pratique

Les exemples pratiques repris dans ce chapitre sont basés sur le scénario décrit ci-dessous. Si vous désirez suivre les étapes individuelles décrites dans les exemples, il est important que vous connaissiez leur objectif principal, et essentiel que vous paramétriez les pages et les conditions comme il est indiqué dans l'exemple repris ici. Notre exemple se focalise sur une situation dans laquelle la plate-forme de gestion de contenu doit fournir trois sites Web. Vu de l'extérieur, les trois sites ont un nom de domaine différent. Le graphisme repose sur une base commune avec de petites touches qui les particularisent.

⁷ James Martin : « Rapid Application Development », Macmillan Publishing Co., Inc., Indianapolis, USA 1991 ; Wilhelm Hasselbring : « Programming languages and systems for prototyping concurrent applications », ACM Comput. Surv. 32(1), 2000, pp. 43-79.

Portail

Le portail présente des éléments de contenu qui ne sont produits, maintenus et archivés que par les rédacteurs compétents. De plus, le portail reprend automatiquement l'information des autres sites présents dans le système.

Sites Web

Le système comporte aussi une série de sites Web édités par d'autres rédacteurs. Les sites de notre exemple sont destinés à différents groupes cibles et sont nommés « B2C » (Business to Customer), « B2B » (Business to Business), et « B2E » (Business to Employee). Chaque rédacteur n'a accès qu'à son interface de travail propre, ainsi qu'à une liste de contenus (voir plus bas) afin de rédiger des actualités au sein d'un workflow prédéfini.

Produits dans les sites Web

En tant qu'unités organisationnelles en dessous du niveau du site Web, nous supposons qu'il existe des zones réservées à la présentation d'un produit particulier. Chaque zone à ce niveau est supervisée par des utilisateurs différents (gestionnaires de produits) .

Listes de contenus/liste de médias

Les éléments de contenu pouvant être utilisés par des rédacteurs de différents sites Web sont enregistrés dans une aire privée ; les rédacteurs peuvent aussi y enregistrer des données destinées à un usage général et/ou à être retravaillées. Les éléments de contenu peuvent être insérés directement à partir de ce dossier à l'aide de l'élément de contenu **Insérer enregistrements**, ou être copiés, selon qu'ils doivent être réédités ou non.

Ceci signifie qu'il est nécessaire d'avoir au moins un groupe pour l'édition du portail, et un groupe pour l'édition de chacun des sites Web. L'assignation de droits au moyen de workflows et d'actions permet de contrôler le flux des modifications apportées par les rédacteurs de chaque site sur les actualités et permet une répartition claire des responsabilités de production et de publication.

4.3.2 Étapes de mise en œuvre

Les étapes suivantes, dans l'ordre spécifié, sont nécessaires à la mise en place du système :

1. Créer une arborescence de base à partir de laquelle les droits peuvent être accordés
2. Créer des groupes qui rassembleront certains utilisateurs
3. Créer des comptes utilisateurs qui seront repris dans un groupe particulier
4. Créer des workflows et des commandes dans l'ordre que vous désirez

La structure de l'arborescence de notre exemple correspond à celle présentée au chapitre 3.10. Dans la prochaine étape, nous créerons les groupes d'utilisateurs ainsi que la base pour les systèmes de droits d'accès.

4.4 Administration des utilisateurs backend

Les rédacteurs et les auteurs travaillent le contenu dans les coulisses du site public (frontend). Leur espace de travail est ce qu'on appelle le backend, dont plusieurs paramètres peuvent être

ajustés en fonction des tâches de chaque participant – on peut même faire en sorte que l'espace de travail soit le frontend.

4.4.1 Créer des groupes d'utilisateurs

Les droits partagés par plusieurs utilisateurs sont définis dans les groupes auxquels ils appartiennent. Les groupes ayant moins de droits deviennent des sous-groupes. On obtient alors une hiérarchie, illustrée par l'exemple suivant :

Groupe A : Point_de_montage a/
Sous-groupe A.1 : Point_de_montage a/1/

Le groupe A.1 hérite de tous les droits du groupe A, en plus de ses propres droits. Si le groupe est membre de plusieurs autres groupes et que certains droits sont contradictoires, il hérite de l'ensemble des droits positifs. Par exemple, si le groupe A n'a pas de droit d'édition pour une certaine page mais que le groupe B l'a, un utilisateur appartenant aux deux groupes aura le droit d'édition.

L'avantage de créer une hiérarchie de groupes réside dans la possibilité de maintenir tous les paramètres utilisateur de base dans un nombre restreint de groupes (en l'occurrence un seul), de sorte que les réglages spécifiques ne doivent être faits que pour un groupe dans l'enregistrement correspondant.

Les groupes d'utilisateurs de notre exemple devraient être créés comme suit :

Groupe « Global »

Contient les paramètres devant s'appliquer à tous les groupes ; de plus, on lui attribue des droits de lecture et d'écriture pour toute l'arborescence fichiers, et les droits d'accès à une section de l'arborescence des pages via les points de montage. Ces points de montage sont définis par les enregistrements **DBmounts**. Dans notre cas, le groupe « Global » contient le point de montage « Contenus ».

Groupe « Portail »

Gère la page portail et est responsable de la publication des nouveaux éléments provenant de tous les niveaux du workflow.

Groupe pour chaque site Web

Ils ne contiennent que le point de montage (DB mount) pour accéder aux sections de l'arborescence du site (« B2C », « B2B » et « B2E ») ainsi qu'un point de montage (File Mounts) pour enregistrer leurs propres fichiers.

Groupe « Produit »

Ce groupe a accès à une partie de l'arborescence de chacun des sites et possède son propre point de montage.

Afin de créer un groupe d'utilisateurs, passez au module **Liste** et cliquez sur le nom de votre installation à côté de l'icône représentant un globe terrestre situé à la tête de votre arborescence. Vous atteindrez ainsi ce qu'on appelle le « niveau racine », qui contient les enregistrements « système ». Vous pouvez alors ajouter un nouvel utilisateur backend à l'aide de la fonction **Créer un nouvel enregistrement**.

Le formulaire est divisé en plusieurs sections :

Data Access

Entrez le nom du groupe ici. Vous pouvez aussi configurer le groupe d'utilisateurs en spécifiant un domaine dans le champ **Lock to domain**, les utilisateurs ne pourront alors s'identifier dans le système que par le biais d'une adresse spécifique (par exemple www.votredomaine.com/typo3). Cette précaution est utile si différents domaines sont configurés sur votre serveur Web ou si vous souhaitez restreindre l'accès TYPO3 à votre intranet.

Access Lists

La section **Access List** contient toutes les options pour la configuration de l'interface backend et pour l'accès à chaque champ de saisie et section de données pour les groupes. En cochant la case **Include Access Lists** le formulaire est rechargé. C'est pourquoi il faut enregistrer toutes les données avant de la sélectionner. Un message d'avertissement s'affiche dès que vous activez l'option.

L'**Access List** contient un menu de sélection qui liste tous les champs de saisie existants ; vous pouvez les sélectionner individuellement en cliquant sur leur nom. Pour sélectionner plusieurs champs, cliquez sur les noms tout en maintenant la touche (ctrl) de votre clavier enfoncée. Les champs de sélection sont les suivants :

Modules

Les éléments listés dans le menu de la colonne de gauche de l'interface backend sont appelés les « modules ». Un rédacteur devrait normalement avoir accès au moins aux menus **Web** → **Page**, **Web** → **Liste**, **Fichier** et **Fichier** → **Fichiers**. Il est utile de laisser à l'utilisateur la possibilité de paramétrer son interface backend, ainsi que la possibilité de modifier son mot de passe via le module **Utilisateur** → **Configuration**.

Figure 4.1:
Après avoir
sélectionné **Include
Access Lists**, la
section **Modules**
apparaît après que le
formulaire a été
rechargé



Tables (listing)

À partir de cette liste, vous pouvez sélectionner les tables de données qui seront visibles pour l'utilisateur. Le nombre de tables présentes varie avec le nombre d'extensions installées, certaines comportant elles-mêmes des tables. C'est pourquoi, lors d'une nouvelle installation, vous devriez vérifier les droits d'accès dans ce champ et le champ suivant.



Figure 4.2:
Sélection des tables
qui seront visibles
pour l'utilisateur

Tables (modify)

Vous sélectionnez ici les tables de la base de données qui seront susceptibles d'être modifiées par l'utilisateur.

Page types

Vous pouvez ici spécifier quels types de pages pourront être modifiés par les membres de ce groupe d'utilisateurs.



Figure 4.3:
Sélection des types de
pages pouvant être
édités par l'utilisateur

Allowed excludefields

À l'aide des champs *Excludefields*, vous pouvez définir plus précisément que dans le menu **Tables** quels champs d'édition seront visibles pour un groupe d'utilisateurs.

Si un champ a été défini en tant qu'« excludefield », et n'est pas sélectionné explicitement dans cette liste, il demeurera invisible pour l'utilisateur membre de ce groupe.

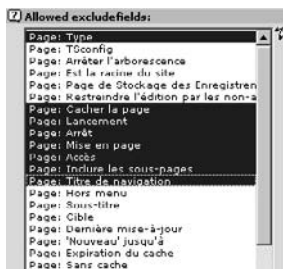


Figure 4.4:
Sélection des champs
d'édition qui seront
visibles pour
l'utilisateur

Explicitly allow/deny field values

Cette option tant attendue a été ajoutée à la version 3.7.0 en réponse au problème de désactivation des types de contenu pour les groupes. Jusqu'alors, on ne pouvait les rendre invisibles que par le biais des options TSConfig. La sélection d'un type de contenu désactive ce dernier pour le groupe concerné. Pour des raisons de sécurité, la désactivation des éléments de contenu « Script » et « HTML » est considérée comme un minimum.

Figure 4.5:
Désactivation de
types de contenu
pour un groupe

2) Explicitly allow/deny field values:

Contenu de la page: Type:

- ☐ ☒ [Interdit] Titre
- ☐ ☒ [Interdit] Texte
- ☐ ☒ [Interdit] Texte & Image
- ☐ ☒ [Interdit] Image
- ☐ ☒ [Interdit] Liste à puces
- ☐ ☒ [Interdit] Tableau
- ☐ ☒ [Interdit] Lien vers fichier
- ☐ ☒ [Interdit] Multimédia
- ☐ ☒ [Interdit] Formulaire
- ☒ ☒ [Interdit] Recherche
- ☒ ☒ [Interdit] Identifiant
- ☐ ☒ [Interdit] Textbox
- ☐ ☒ [Interdit] Menu / Plan site
- ☐ ☒ [Interdit] Insérer enregistrements
- ☒ ☒ [Interdit] Insérer un plugin
- ☒ ☒ [Interdit] Script
- ☐ ☒ [Interdit] Séparation
- ☒ ☒ [Interdit] HTML

Sélectionner toutes les cases à cocher

Contenu de la page: Plugin:

- ☒ ☒ [Interdit] Nouvelles
- ☒ ☒ [Interdit] Adresses
- ☐ ☒ [Interdit] General Office Display

Sélectionner toutes les cases à cocher

Limit to Languages

Ici, on peut limiter l'accès des groupes à certaines langues du site.

Custom Module Options

Les options spécifiques aux modules d'extension backend peuvent s'insérer dans cette section.

DB Mounts

Un point de montage de type **Database (DB) Mounts** permet d'assigner à l'utilisateur un point d'entrée dans l'arborescence des pages. Si l'utilisateur appartient à plusieurs groupes différents, et qu'un point d'entrée est défini plusieurs fois, il apparaîtra le même nombre de fois dans le backend de l'utilisateur. C'est pourquoi vous devez veiller à ne pas créer de paramètres se répétant dans plusieurs groupes. Dans notre exemple, vous devriez configurer les groupes comme suit :

Groupe	Point de montage
Global	Contenus
Portail	Portail
Produits	Produits
B2C	B2C Accueil
B2B	B2B Accueil
B2E	B2E Accueil

Tableau 4.1:
Exemple de groupes
d'utilisateurs et leurs
points de montage

Filemounts

Ce champ fait référence aux répertoires du système de fichiers qui sont assignés aux groupes d'utilisateurs dans lesquels les utilisateurs peuvent enregistrer leurs fichiers. Vous devez sélectionner un point de montage de type fichiers. En tant qu'enregistrements « système », ces points de montage sont sauvegardés au niveau racine de l'installation TYPO3, tout comme les utilisateurs et les groupes. On peut soit les créer à cet endroit, soit y accéder dans le formulaire d'édition de groupe via les icônes **Edit filemount**, **Create filemount** ou **List filemount** situés à droite du champ de sélection **File Mounts**.

Afin de définir un point de montage, l'administrateur doit avoir créé les sous-répertoires correspondants dans le répertoire **fileadmin**, via le module **Fichier** → **Fichiers**. Les noms des répertoires dans le système de fichiers et les noms des points de montage ne sont pas nécessairement les mêmes. Dans notre exemple, nous donnerons les noms suivants aux répertoires déjà existants :

Groupe	Nom de point de montage	Chemin
Global	Media	Media/
Portail	Tous les fichiers	fichiers/
B2C	Fichiers B2C	fichiers/fichiers_B2C/
B2B	Fichiers B2B	fichiers/fichiers_B2B/
B2E	Fichiers B2E	fichiers/fichiers_B2E/
Produit	Fichiers P	fichiers/fichiers_p/

Tableau 4.2:
Exemples de groupes,
points de montage et
chemins

Le nom qui sera visible pour les utilisateurs est entré dans le champ **Label**. Dans le champ **Path**, spécifiez le chemin à partir du répertoire **fileadmin** situé dans le répertoire Web de votre serveur Web. Vous pouvez aussi spécifier un chemin absolu. Dans les deux cas, le nom du chemin doit se terminer par une barre oblique.

Hide in Lists

Cette option sert à supprimer l'affichage du groupe dans le module **Utilisateur** → **Centre de tâches** ainsi que dans le module **Web** → **Accès**. De cette façon, on peut éviter que les utilisateurs envoient des messages (dans le module **Utilisateur**

→ **Centre de tâches**) à des groupes globaux qui n'ont été créés qu'à des fins administratives.

Subgroups

L'option **Subgroups** permet d'assigner le groupe en tant que sous-groupe d'un autre groupe dont il hérite des droits et des paramètres. Dans notre exemple, nous définirons le groupe « Global » comme ayant pour sous-groupes tous les autres groupes, afin que ces derniers en reçoivent toute la configuration.

TSCnf

Le dernier champ du formulaire permet d'insérer du code TypeScript. Cette section nommée TSCnf est vue plus en détail à la section 4.8.

Vous pouvez maintenant enregistrer et fermer le formulaire à l'aide du bouton correspondant. Si vous avez utilisé l'option **Include Access Lists** lors de la création d'un groupe, ce dernier sera affiché avec une icône rouge dans l'affichage liste (module **Liste**). Les groupes d'utilisateurs n'ayant pas été édités à l'aide de cette dernière option reçoivent une icône bleue.

4.4.2 Créer des comptes utilisateurs

Après avoir créé des groupes, c'est au tour des utilisateurs, qui peuvent être assignés à ces groupes. Le formulaire de création des comptes utilisateurs ne diffère de celui de la création de groupes que par les options pour la configuration des accès aux fichiers dans le module **Fichier**.

Les utilisateurs devraient toujours avoir leur propre compte utilisateur car celui-ci peut être utilisé non seulement pour l'attribution de droits, mais aussi à des fins administratives ou de travail collaboratif.

- Les actions des utilisateurs peuvent être tracées avec une fonction log
- Les utilisateurs peuvent communiquer via le module **Utilisateur** → **Centre de tâches** en utilisant de simples fonctions de travail collaboratif (Actualités, Tâches, Notes).
- Les utilisateurs peuvent régler eux-mêmes l'affichage et le mode de fonctionnement du backend, ou bien recevoir un environnement de travail personnalisé par l'administrateur.

Dans la prochaine étape, nous ajouterons un utilisateur à chaque groupe, afin de pouvoir ensuite tester la configuration. De retour au niveau racine du système, sélectionnez **Créer un nouvel enregistrement** et **Utilisateur backend**. Voici les différentes sections du formulaire :

Données d'accès

La première section du formulaire contient les champs pour l'introduction d'un nom d'utilisateur, d'un mot de passe, d'une affectation à un groupe, et l'option pour lier un utilisateur à un nom de domaine.

Les noms d'utilisateurs ne doivent contenir que des lettres minuscules, les espaces ne sont pas admis. Lorsque vous entrez un mot de passe, il demeure visible jusqu'à ce que vous l'ayiez sauvegardé pour la première fois ; il est ensuite enregistré sous un format de hachage MD5 dans la base de données et est transféré dans le backend sous cette forme lors de l'identification via le serveur Web. Ceci a pour conséquence que le mot de passe oublié ne peut être relu : il doit alors être réassigné.

Groupes d'utilisateur

On peut affilier l'utilisateur à un ou plusieurs groupes via le champ de sélection. Les options situées à droite du champ permettent de créer, d'éditer ou de lister des groupes d'utilisateurs. L'ordre dans lequel les groupes sont mentionnés est important lors de la définition des droits. Les pages nouvellement créées par l'utilisateur appartiennent toujours au premier groupe apparaissant dans la liste. On peut modifier ce paramètre à l'aide de TSConfig, tout comme les droits des utilisateurs, des groupes et de tous les autres. Ceci est illustré à la section 4.8. L'option **Lock to Domain** garantit, dans des systèmes comportant plusieurs domaines Internet, que les utilisateurs ne s'identifient que dans leur propre domaine.

Admin

La seconde section du formulaire contient l'option **Admin**, qui donne à l'utilisateur un accès illimité au système. Autant que possible, cette configuration ne devrait être appliquée que pour un seul utilisateur dans le système, ou du moins être utilisée le moins souvent possible. A quelques exceptions près, un utilisateur ayant les droits d'un administrateur est en mesure de détruire irrémédiablement la configuration et tous les éléments de contenu.

Données utilisateurs

La section suivante sert à la saisie de données sur l'utilisateur ; l'utilisateur peut les modifier via le module **Utilisateur** → **Configuration**, bien qu'il n'ait pas complètement accès à l'enregistrement.

DB Mounts et Filemounts

Le système de filemounts et DB mounts a déjà été décrit à la section 4.4.1. Lorsque les options **DB Mounts** et **File Mounts** sont désactivées dans la section **Mount from groups**, l'utilisateur n'hérite plus des paramètres du groupe auquel il appartient en ce qui concerne le point de montage. De plus, cette action peut influencer les droits disponibles pour l'utilisateur en termes de fichiers dans son propre point de montage.

TSConf

Les possibilités de configuration de l'utilisateur via TypoScript sont vues plus en détail à la section 4.8.

Vous pouvez maintenant enregistrer et fermer le formulaire : l'utilisateur est créé. Si vous quittez le système et que vous vous identifiez maintenant en tant qu'un des nouveaux utilisateurs, vous pourrez vérifier votre configuration.

Exemple

Créez les utilisateurs suivants pour les groupes « Portail », « B2C », « B2B » et « B2E » :

Utilisateur	Groupe
Portail-redacteur	Portail
b2c-redacteur-1	B2C
b2c-redacteur-2	B2C
b2b-redacteur	B2B
b2e-redacteur	B2E

Tableau 4.3:
Exemple d'utilisateurs

La création de comptes utilisateurs pour les gestionnaires de produits dont nous avons parlé ci-dessus sera vue plus tard, en même temps que la fonctionnalité des commandes (cf. section 4.10).

4.5 Administration des utilisateurs à l'aide du module Outils → Administration des utilisateurs

Le module **Outils** contient une interface d'administration et d'analyse, dans le sous-module **Administration des utilisateurs**, qui joue un rôle primordial dans le travail quotidien de l'administrateur. Cet outil permet d'afficher l'état actuel des droits à l'aide de plusieurs critères, et d'effectuer des modifications si nécessaire.

La configuration d'un utilisateur ou d'un groupe (si vous avez préalablement sélectionné l'affichage de groupes) s'affiche individuellement lorsque vous cliquez sur le nom d'utilisateur ou de groupe. On peut entre autres y voir la liste des pages auxquelles l'utilisateur en question n'a pas accès. Les autres paramètres sont expliqués dans le tableau ci-dessous. Les options pour éditer les enregistrements fonctionnent de la même façon que celles du module **Liste**.

L'option **SU** de la liste d'utilisateurs offre une option particulière. En cliquant sur cette icône (SU pour *Switch User* – comparable à la commande UNIX du même nom), l'administrateur passe dans le compte utilisateur sélectionné. Cette option est très utile pour vérifier les configurations. Dans ce cas toutefois, la seule façon de retourner à l'interface administrateur est de sortir de TYPO3 et de s'identifier à nouveau en tant qu'administrateur. L'administrateur peut ainsi se glisser dans n'importe quel profil d'utilisateur, peu importe le cas, même s'il n'en connaît pas le mot de passe.

L'affichage des utilisateurs et des groupes dans l'aperçu d'analyse peut être paramétré par les options suivantes, pouvant être utilisées seules ou combinées :

Tableau 4.4:
Affichage
d'utilisateurs et de
groupes

Option	Signification
Filemounts	Comparaison des utilisateurs en fonction des répertoires auxquels ils ont accès
Webmounts	Comparaison des utilisateurs en fonction des pages de l'arborescence auxquelles ils ont accès
Default upload path	Chemin pour les téléchargements de fichiers à partir d'éléments de page
Main user group	Premier groupe dont l'utilisateur est membre
Member of groups	Autres groupes dont l'utilisateur est membre
Page types access	Types de page que les utilisateurs peuvent créer
Select Tables	Tables de la base de données que l'utilisateur peut visualiser
Modify Tables	Tables de la base de données que l'utilisateur peut modifier

suite

Option	Signification
Non-exclude fields	Champs que l'utilisateur peut éditer
Explicit Allow/Deny	Éléments de contenu que l'utilisateur n'est pas autorisé à insérer
Limit to languages	Langues dans lesquelles les membres d'un groupe peuvent ou ne peuvent pas éditer des éléments de contenu
Custom Module Options	Configuration de droits ajoutés par un module d'extension backend
Modules	Modules backend auxquels l'utilisateur a accès
TSGlobal	Paramètres TSGlobal pour cet utilisateur
TSGlobal HL	Paramètres TSGlobal pour cet utilisateur en mode étendu

Si vous avez saisi les valeurs mentionnées dans les sections précédentes, l'utilisateur admin apparaît comme suit, après que vous avez coché les cases **Main User Group** et **Member of Groups** :

RESULT					
Usernames:		Main user group:		Member of groups:	
 admin	  				
 b2b-redacteur	  	 Groupe b2b	 	 Global	 
 b2c-redacteur-1	  	 Groupe b2c	 	 Global	 
 b2c-redacteur-2	  	 Groupe b2c	 	 Groupe b2c	 
 b2e-redacteur	  	 Groupe b2e	 	 Global	 
 b2e-redacteur	  	 Groupe b2e	 	 Groupe b2e	 
 portail-redacteur	  	 Groupe Portail	 	 Global	 
		 Groupe Portail	 	 Groupe Portail	 

Figure 4.6:
Affichage de tous les groupes dont l'utilisateur est membre (droite) et du premier groupe dont il est membre (centre)

Grâce à cette distinction entre les utilisateurs, il est facile de maintenir des paramètres via des groupes de base communs (« Global »), tout en assignant aux utilisateurs individuels des environnements de travail complètement différents ainsi que des sections de l'arborescence séparées. Afin de vérifier la configuration de chacun des groupes d'utilisateurs et de les peaufiner, vous pouvez aussi passer dans chacun des comptes utilisateurs grâce à **SU** et vérifier la gamme de fonctions disponibles pour l'utilisateur en question.

ATTENTION : pour qu'un utilisateur soit en mesure d'éditer des pages, ces dernières doivent être rendues accessibles à l'aide du module décrit ci-dessous.

4.6 Droits d'accès au niveau de la page

Comme dans le système de fichier sous Linux, le module d'accès vous permet d'accorder des droits aux « utilisateurs », aux « groupes » et à « tous les autres ». Au départ, ces droits sont

fixés automatiquement lors de la création d'une page, peu importe le créateur de cette page. Le créateur en devient automatiquement propriétaire, et le premier groupe auquel il appartient obtient les droits d'accès. Ces valeurs, ainsi que les permissions accordées automatiquement, peuvent être modifiées et ajustées de façon individuelle via TSConf (cf. section 4.8).

Lors de son ouverture, le module **Web** → **Accès** se réfère toujours à la page sélectionnée, affichée dans le coin supérieur gauche de la vue détaillée. Le mode **Utilisateur (vue d'ensemble)** est alors activé. Ce mode d'affichage montre les pages en une arborescence qui reprend le nombre de niveaux pouvant être configurés. Le mode **Permissions** affiche les pages avec les droits accordés au propriétaire, au groupe et à tous les autres.

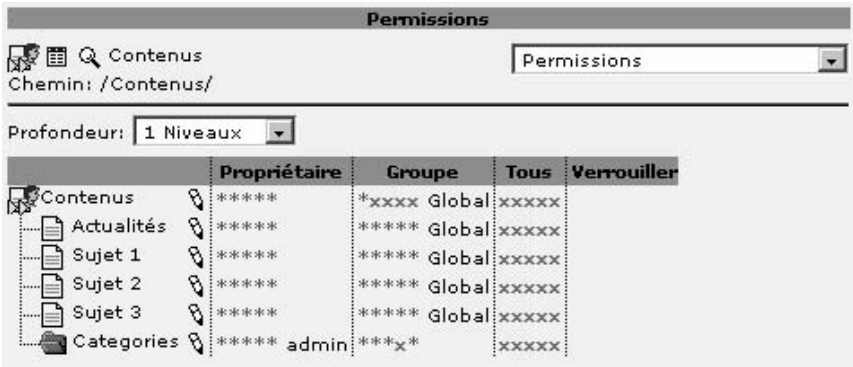
Une fois que vous avez sélectionné une page, un formulaire s'affiche. En cliquant sur l'icône « crayon », vous pouvez attribuer les droits de cette page au propriétaire et aux groupes. Vous pouvez ensuite déterminer jusqu'à quel niveau de sous-pages ces paramètres sont valables.

Exemple

Passez au module **Accès** et sélectionnez la page « Contenus » dans l'arborescence. Ouvrez ensuite le formulaire d'édition en cliquant sur l'icône crayon. Assignez cette page ainsi que ses sous-pages au groupe utilisateur « Global » et accordez les permissions d'accès comme illustré dans la capture d'écran 4.7. Lorsque vous enregistrez vos entrées, vous êtes ramené à l'aperçu utilisateur. A présent, ouvrez de nouveau le formulaire d'édition de la page et rééditez les droits d'accès seulement pour cette page, en ramenant le nombre de niveaux à celui de la page actuelle. Vous avez ainsi créé une situation dans laquelle les utilisateurs d'un groupe peuvent éditer toutes les sous-pages, mais pas la page racine, sans que l'administrateur ne doive intervenir pour modifier la configuration de chacune des pages.

Le résultat final devrait maintenant ressembler à ceci :

Figure 4.7:
Affichage des droits
d'accès dans le
module Web → accès



Afin de rendre toutes les sections de l'arborescence disponibles pour tous les utilisateurs, il est plus simple d'accorder tous les droits correspondant au groupe « Global », excepté pour les pages d'« Accueil » et toutes les pages ne devant pas être modifiées. Notez que les accès sont déjà restreints dans l'arborescence par les points de montage définis pour chacun des groupes.

4.7 Édition frontend pour utilisateurs backend

Référence 884598

Nous n'avons pas encore mentionné l'un des plus grands avantages de TYPO3 : il est possible pour les utilisateurs de travailler directement sur le site, sans avoir recours au backend, ou comme alternative à ce dernier. TYPO3 supporte par défaut la possibilité d'éditer les éléments de contenu après l'affichage d'une page, ainsi que d'ajouter de nouvelles pages. À l'aide d'une simple extension, les utilisateurs concernés peuvent en outre être redirigés directement dans le site Web (dénommé le *frontend*), voire travailler uniquement en tant qu'utilisateurs frontend (« frontend-only »).

Une troisième possibilité consiste à guider le rédacteur via un hyperlien d'une autre page à la page d'identification vers le backend, et d'ajouter un paramètre à ce lien qui le ramènera, après son identification, dans le site Web avec options d'édition, plutôt que dans le backend. Le backend demeure ainsi accessible pour le rédacteur si nécessaire.

Un lien ramenant l'utilisateur à la page d'accueil du site après l'identification, ressemble à ceci :

```
<a href="typo3/index.php?redirect_url=../">Identification au backend avec
redirection vers le frontend</a>
```

Le **Panneau d'administration** sert d'outil d'administration du module d'édition frontend. Il offre les fonctionnalités d'édition requises par une interface utilisateur simplifiée. Bien sûr, le panneau d'administration et ses options peuvent être personnalisés pour chaque utilisateur.

Les administrateurs sont automatiquement autorisés à utiliser l'*édition frontend*. Afin d'en donner aussi la possibilité à un utilisateur, la configuration suivante est nécessaire :

1. L'affichage du panneau d'administration dans le frontend du site Web doit être configuré. Entrez la commande suivante dans le champ **Setup** du gabarit de votre page :

```
config.admPanel = 1
```

2. Afin de donner maintenant des accès de groupe aux options d'édition frontend ainsi qu'un panneau d'administration à des utilisateurs individuels, la configuration TypoScript suivante doit être ajoutée dans le champ **TSCConf** du groupe de l'utilisateur. Pour y parvenir, entrez la commande suivante directement dans le champ **TSCConf** ou utilisez l'assistant :

```
admPanel {
    enable.edit = 1
}
```

Mais les options peuvent être réglées de façon encore plus précise. En ce qui concerne des rédacteurs particuliers, on peut utiliser la configuration suivante, qui affiche automatiquement toutes les options d'édition mais garde le panneau d'administration invisible :

```
admPanel {
    enable.edit = 1
    module.edit.forceDisplayIcons = 1
    module.edit.forceDisplayFieldIcons = 1
    hide = 1
}
```

On peut accéder individuellement à chaque section du panneau d'administration. Par exemple, même les options du cache peuvent être affichées pour un rédacteur à l'aide de la commande :

```
enable.cache = 1
```

Le panneau d'administration se compose des sections suivantes, accompagnées de leurs clés TypoScript :

Tableau 4.5:
Sections du panneau
d'administration et
les clés TypoScript

Nom	Clé TS	Fonction
Prévisualiser	preview	Options de prévisualisation
Cache	cache	Vider le cache et options du cache
Publier	publish	Options pour l'export statique de pages HTML
Éditer	edit	Options d'édition
TypoScript	tsdebug	Différentes fonctions pour le développement de gabarits, surtout pour le débogage
Info	info	Informations sur la page

Chaque partie du panneau d'administration peut être activée séparément, mais vous pouvez aussi l'afficher entièrement :

```
enable.all = 1
```

plutôt que de devoir activer chaque section séparément.

L'édition frontend ne se limite pas qu'aux pages et aux éléments de contenu. En principe, n'importe quelle table de la base de données peut être configurée pour l'édition frontend à l'aide des entrées correspondantes dans le gabarit de la page. De cette façon, l'édition frontend peut aussi être utilisée si des éléments de contenu tels que des enregistrements de produits, d'actualités, ou d'autres contenus assignés aux pages doivent être édités.

Voici un exemple TypoScript qui configure l'édition de nouvelles entrées de la table de base de données `tt_news` :

```
styles.content.editPanelPageRight = COA
styles.content.editPanelPageRight {
    10 = EDITPANEL
    10 {
        newRecordFromTable = tt_news
        allow = toolbar,edit,move,hide,delete,new
        label = page:<B>%s</B><br>&nbsp;créer une actualité
        edit.displayRecord = 1
        line = 4
    }
}
```

Si vous insérez cet objet avec l'expression

```
page.20 < styles.content.editPanelPageRight
```

dans le gabarit d'une page, les options d'édition grâce auxquelles les actualités peuvent être éditées sur cette page s'affichent.

4.8 TSConfig – options et interface

TypoScript ne sert pas seulement à écrire des gabarits. Avec la même syntaxe – excepté pour les paramètres du type *Constants* et *Conditions* qui n'existent pas dans ce cas-ci – les valeurs peuvent être définies de façon similaire à la base de registre de Windows (Windows Registry). On peut agir sur deux plans :

TSConfig utilisateur

Par utilisateur et par groupe, TSConfig peut être utilisé pour influencer globalement l'affichage du backend, ou seulement les modules individuels du backend.

TSConfig page

Au niveau de la page, TSConfig peut être utilisé afin de configurer différentes sections du site Web.

Un paramètre **TSConfig utilisateur** peut être utilisé en remplacement d'une option spécifique d'une arborescence, afin de permettre par exemple à un administrateur d'avoir un affichage différent de celui d'un utilisateur dont l'espace de travail est plus restreint.

Des propriétés **TSConfig page** entrées dans l'en-tête d'une page s'étendront toujours à toutes ses sous-pages. Afin de permettre aux administrateurs de travailler rapidement avec ce système, un outil a été développé, appelé *TypoScript Property Lookup Wizard*.

4.8.1 Assistant TSConfig : consulter les propriétés TypoScript

Lorsque vous cliquez sur l'icône **TS** située à droite du champ de saisie dans le formulaire d'édition de l'en-tête d'une page, d'un utilisateur ou d'un groupe, une aide en ligne s'ouvre et affiche toutes les options TSConfig disponibles. Cette référence en ligne est disponible pour toutes les installations TYPO3, et l'assistant TSConfig vous permet de transférer des valeurs TSConfig directement dans le champ de saisie TSConfig du formulaire.

L'assistant est très simple : un clic sur une valeur transfère directement cette dernière dans la fenêtre TSConfig du formulaire d'édition, tout en fermant l'assistant.

Si vous cliquez sur « + » au lieu de cliquer sur le nom, les paramètres sont transférés dans le champ de saisie de l'éditeur. L'option **Entourer** place le paramètre entre crochets. **Indenter** et **Désindenter** servent à indenter les lignes, afin d'apporter plus de clarté.

4.8.2 TSConfig utilisateur

Pour chaque groupe d'utilisateurs, TypoScript offre une série de configurations à l'aide de TSConfig. Vous obtenez un aperçu si vous ouvrez l'assistant à partir du formulaire d'édition d'un compte utilisateur ou d'un groupe. Tout comme l'arborescence d'objets au niveau des gabarits, nous appelons cet aperçu l'arborescence **TSConfig utilisateur** ou **TSConfig page**.

Les différentes sections de l'arborescence TSConfig utilisateur correspondent aux options de configuration suivantes :

Référence 198531

admPanel

Configuration du panneau d'administration du frontend

options

Configuration générale du backend

setup

Contient les branches **default** et **override** ; les options de configuration dans le module **Utilisateur** → **Configuration** peuvent aussi être contrôlées ici avec TypoScript. Ces deux branches contiennent les mêmes paramètres, mais ont un effet différent. Avec **default**, vous pouvez, tout comme l'administrateur, influencer les réglages par défaut que l'utilisateur retrouve dans le module. Si l'utilisateur choisit pour ce module de restaurer la configuration par défaut, le système retourne aux paramètres configurés par **default**. Le paramètre **override** permet de prédéfinir pour les utilisateurs les paramètres qui ne peuvent être modifiés dans le module **Utilisateur** → **Configuration**. Ceci peut s'avérer utile du point de vue de l'administrateur, afin d'éviter, par exemple, que des pages ne soient éventuellement effacées récursivement.

Mod

La section **Mod** fait référence aux sous-modules du module **Web**. Une série de paramètres sont disponibles afin de configurer ces sous-modules.

ATTENTION : les options de configuration pour le module **Info** ne sont actuellement pas actives !

Exemple

Tout d'abord, fixons différentes options de visualisation pour tous les groupes, grâce à la configuration du groupe « Global ».

Parce que la mise en page n'utilise que les colonnes **normal** et **droite**, l'affichage des colonnes **gauche** et **bordure** dans le backend est inutile. On sélectionne donc les colonnes avec l'entrée suivante :

```
mod {  
    SHARED.colPos_list = 0,2  
}
```

Les colonnes sont répertoriées comme suit : gauche=1, normal=0, droite=2, bordure=3.

Afin d'obtenir un affichage uniforme pour tous les utilisateurs de ce groupe, vous devez, par exemple, désactiver l'affichage de vignettes dans le backend, et faire en sorte que l'utilisateur soit redirigé vers le module **Utilisateur** → **Centre de tâches** après identification, plutôt que vers la page d'aide :

```
setup.defaults {  
    thumbnailsByDefault = 0  
    startInTaskCenter = 1  
}
```

Les paramètres pour les utilisateurs peuvent être visualisés et comparés dans le module **Outils** → **Administration des utilisateurs** avec les options **TSCConfig** et **TSCConfig HL**. **TSCConfig**

affiche le même contenu que le navigateur **Configuration TS de la page** du module **Info**. **TSConfig HL** met en évidence, par l'emploi de couleurs, les valeurs et la syntaxe.

4.8.3 TSConfig page

D'autres options s'ajoutent à celles du **TSConfig utilisateur** au niveau de la page. Les options suivantes peuvent aussi être assignées aux utilisateurs et aux groupes dans leurs champs TS-Config. Inversement, les options **TSConfig utilisateur** ne peuvent pas être entrées dans les pages. Au niveau de la page, les sections suivantes sont disponibles :

Référence 168197

TSFE

Contient une option permettant de transférer une session utilisateur.

mod

Contrôle les menus des modules backend.

TCEMAIN

Concerne les options qui peuvent être configurées pour chaque table de système, telles que celles reprenant le nombre d'entrées et les limites de temps pour l'historique d'édition ; cette option permet aussi d'accorder des droits pour une arborescence sans tenir compte de la configuration des utilisateurs et des groupes pour la création de pages.

TCEFORM

Fait référence à la configuration dans les formulaires du backend et des champs que ceux-ci contiennent.

RTE

Le *Rich Text Editor* peut aussi être paramétré dans TSConfig. Cependant, on n'utilise généralement pas toutes les possibilités car plusieurs options ne sont plus compatibles avec les standards actuels du Web.

Exemples TCEMAIN

TCEMAIN permet de configurer les droits des utilisateurs ou des groupes pour la création de nouvelles pages. L'entrée suivante dans la page d'accueil d'une arborescence,

```
TCEMAIN.permissions.groupid = 1
```

spécifie que de nouvelles pages créées dans cette arborescence appartiendront automatiquement au groupe « Global » plutôt qu'au groupe utilisateur principal. Le groupe est identifié par son UID (« 1 » dans notre exemple) dans la base de données. L'UID est aussi affiché par le module **Web** → **Liste** du backend, si vous cliquez sur l'icône d'information dans l'affichage liste en mode étendu ou si vous placez le curseur sur l'icône du groupe dans l'affichage liste).

La même entrée dans le groupe « Global », dont tous les autres groupes font partie, permet à ces derniers d'éditer toutes les pages, pour autant qu'ils y aient accès. De cette manière, les utilisateurs d'un groupe (ou l'administrateur) peuvent créer des pages et ne doivent plus spécifier les permissions pour que les utilisateurs d'un autre groupe puissent y accéder.

Après avoir assigné de nouveaux éléments de façon permanente à un groupe, les droits de ce groupe sur les pages peuvent être configurés. On utilise alors la clé **permissions.group** :

```
TCEMAIN.permissions.group = show, editcontent
```

Les valeurs disponibles sont **Show** (afficher dans le backend), **Editcontent**, **Edit** (éditer l'en-tête de la page) , **New** (créer de nouvelles pages), et **Delete**.

Lorsque vous copiez ces éléments dans le backend de TYPO3, des suffixes sont automatiquement ajoutés aux noms. « Page1 » devient alors « Page1 (copy) » une fois qu'elle a été copiée et insérée quelque part. Cette fonction peut être désactivée en entrant :

```
TCEMAIN.default.disablePrependAtCopy = 0
```

Référence 036769 La liste exhaustive des options TCEMAIN se trouve à la référence ci-contre.

Référence 176791 Vous pouvez trouver une liste des clés TypoScript pour paramétrer les tables à la référence ci-contre.

Exemples TCEFORM

On peut paramétrer tous les formulaires d'entrée en ce qui concerne l'affichage et le nom des options grâce à **TCEFORM**. On utilise alors la notation suivante :

```
TCEFORM.[nom_table].[nom_champ].[clé_TSConf] = valeur.
```

Référence 538296 Les clés TSConf ainsi que les valeurs sont définies à la référence ci-contre.

Tout d'abord, pour des raisons de sécurité, nous retirons l'accès à certains types de contenu pour notre groupe d'utilisateurs « Global » :

```
TCEFORM.tt_content.CType.removeItem = search, login, div, script, html
```

Les éléments **script** et **HTML** devraient toujours être supprimés dans chaque installation, parce qu'ils représentent un risque important au niveau de la sécurité entre les mains de pirates informatiques expérimentés ayant réussi à accéder au backend. Ils demeurent toutefois disponibles dans l'assistant, qui s'affiche automatiquement si vous sélectionnez l'option **Créer du contenu de page**. Nous supprimerons donc l'affichage de cet assistant pour le groupe « Global » à l'aide de l'entrée suivante :

```
mod.weblayout.disableNewContentElementWizard = 1
```

Dans la configuration par défaut, (sans extensions, qui étend **tt_content**), les éléments suivants peuvent être sélectionnés, et sont ajoutés à l'aide de l'expression contenue dans la colonne « clé TS » :

Tableau 4.6:
Éléments des
formulaires d'entrée

Français	Anglais	clé TS
En-tête	Header	header
Texte	Text	text
Texte & Image	Text w/image	textpic
Image	Image	image

suite

Français	Anglais	clé TS
Liste à puces	Bullet list	bullets
Tableau	Table	table
Lien vers fichier	Filelinks	uploads
Multimédia	Multimedia	multimedia
Formulaire	Form	mailform
Recherche	Search	search
Identifiant	Login	login
Textbox	Textbox	splash
Menu/Plan du site	Menu/Sitemap	menu
Insérer enregistrements	Insert record	shortcut
Insérer un plugin	Insert plugin	list
Script	Script	script
Séparation	Divide	div
HTML	HTML	html

Vous pouvez aussi spécifier un autre nom pour chacun des éléments. Vous devez alors connaître le nom de la table et du champ tels qu'ils sont entrés dans la base de données. Les noms des champs sont repris dans la structure de base de données, que vous pouvez afficher via le module **Outils** → **Configuration**. Le tableau **\$TCA**, dans le sous-item **tt_content**, contient tous les champs de la table de base de données dans lesquels on peut enregistrer du contenu de page.

Les en-têtes peuvent être remplacés par vos propres noms à l'aide de la notation suivante :

```
TCEFORM.[nom_table].[champ_table].[position] = valeur
```

La ligne TypeScript suivante renomme donc l'en-tête standard « Normal » du champ **header type** en « centré/en haut » :

```
TCEFORM.tt_content.header_layout.altLabels.0 = centré/en haut
```

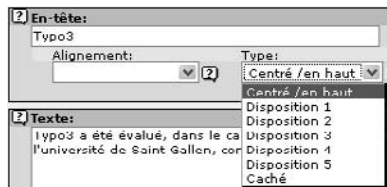


Figure 4.8:
Options
personnalisées dans
le menu d'en-tête

L'option **removeItems** permet d'exclure complètement certaines valeurs des listes de sélection :

```
TCEFORM.tt_content.menu_type.removeItem = 3,4,5
```


Le nombre entier désigne la position dans le menu de sélection. La ligne ci-dessus cache donc les entrées placées en troisième, quatrième et cinquième position de l'élément de contenu « Menu/Sitemap ».

4.8.4 Ajustement du Rich Text Editor

Le *Rich Text editor* (RTE) est l'éditeur WYSIWYG du système TYPO3, qui fournit aux utilisateurs différentes options de mise en forme et plusieurs assistants, basés sur la technologie d'Internet Explorer. En principe, toutes les options offertes par Internet Explorer sont incluses, même si la plupart de ces options sont en pratique incompatibles avec les principes de conception standards de mise en page HTML. En règle générale, la gamme de fonctions du RTE est fortement restreinte et peut être ajustée à vos propres spécifications de mise en forme, grâce aux options de configuration du RTE.

Le Rich Text Editor propose essentiellement les interfaces de configuration suivantes :

1. Configuration des champs d'entrée pour lesquels le RTE devrait être disponible ; le RTE peut avoir une configuration différente pour chaque champ.
2. Possibilité d'activer ou de désactiver les options d'édition dans la barre de menu.
3. Configuration des options pour la mise en forme des paragraphes et des caractères.
4. Paramètres de transformation pour les entrées dans le RTE lors de la sauvegarde ainsi que pour la publication à partir de la base de données.

Configuration RTE restrictive

Notre premier exemple se limite à une variante très simple, pour les raisons déjà évoquées. D'après notre expérience, cette configuration convient dans la plupart des cas.

Le RTE peut être activé ou désactivé pour certaines sections de l'arborescence via le champ correspondant du **TSConfig page**.

Dans un premier temps, nous configurons le RTE pour qu'il ne soit disponible que pour les éléments de contenu **Texte** et **Texte & image**, même si des extensions utilisant le RTE sont installées :

```
RTE.default.disabled = 1
RTE.config.tt_content.bodytext.types {
    text.disabled = 0
    textpic.disabled = 0
}
```

Ensuite, nous spécifions que toutes les options, en-têtes et autres mises en formes existantes ne doivent plus être incluses dans le champ texte même, mais que seules les mises en forme standards (afin de faciliter la restauration pour les rédacteurs) et la mise en forme pour les citations doivent être disponibles. La suppression des en-têtes dans les champs du RTE est particulièrement utile, parce qu'elle force les rédacteurs à définir le texte en plusieurs sections dans des éléments de contenu séparés, ce qui améliore la clarté de la rédaction et permet à d'autres options, telles que des menus de section, d'entrer en jeu.

```

RTE.classes {
  highlight {
    name = surbrillance
    value = font:bold; color:navy;
  }
  citation {
    name = citation
    value = font:italic 15px; margin-left:20px;
  }
}
RTE.default.classesCharacter = highlight, citation

```

Dans **TSConfig utilisateur**, vous pouvez seulement configurer l'affichage des menus du RTE pour chacun des groupes ou des utilisateurs. Les options configurées ici sont disponibles pour tous les membres du groupe « Global », ce qui correspond à tous les utilisateurs du système, à l'aide de l'entrée suivante dans le champ **TSconf** du groupe :

```
options.RTEkeyList = class, bold, italic, link
```

Les rédacteurs voient alors la barre de menu suivante s'afficher (figure 4.7).

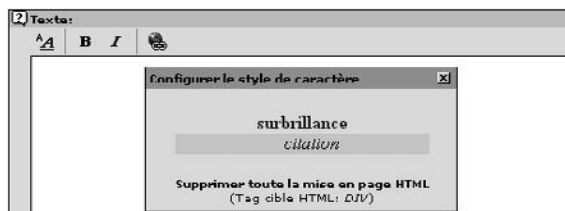


Figure 4.9:
Barre de menu réduite
avec mise en forme
personnalisée

Comme vous pouvez le constater, les options pour insérer des tableaux ou des images via le menu ont aussi été supprimées. Pour y pallier, il existe des éléments de contenu prévus à cet effet, plus simples à contrôler pour obtenir une apparence uniforme, et préférables pour des questions de clarté.

Configuration étendue

Le Rich Text Editor possède plusieurs options qui ne sont pas encore visibles dans l'affichage par défaut, et doivent être ajoutées. Elles comprennent un objet utilisateur avec lequel vous pouvez ajouter vos propres définitions.

À l'aide de l'entrée suivante – par exemple dans le champ **TSConf** de l'administrateur système – nous pouvons activer tous les éléments disponibles du RTE :

```

options.RTEkeyList = cut, copy, paste, formatblock, class, fontstyle,
fontsize, textcolor, bold, italic, underline, left, center, right,
orderedlist, unorderedlist, outdent, indent, link, table, bgcolor,
image, emoticon, line, user, chMode

```

Ci-dessous se trouve un exemple de configuration de classes :

```
RTE.default {
  colors = color1, color2, noColor
  PROC.allowedClasses = left, right
  PROC.allowTagsOutside = IMG
  mainStyle_font = Verdana, sans-serif
  mainStyle_size = 12px
  mainStyle_color = #313031
  mainStyleOverride_add.P = font-family : Verdana,
  sans-serif; font-size : 12px;
  mainStyleOverride_add.H1 = font-family : Verdana,
  sans-serif; font-size : 18px;
  mainStyleOverride_add.H2 = font-family : Verdana,
  sans-serif; font-size : 12px;
  inlineStyle_img = margin: 5px;
  hidePStyleItems = H3, H4, H5, H6, pre
  classesImage = middle, withoutmargin
  classesCharacter = red, middle, small, large,
  gray, left
  classesParagraph = left
}

RTE.classes {
  withoutmargin.name = Normal, without margin
  withoutmargin.value = margin: 0;
  red.name = red
  red.value = color: red;
  middle.name = middle
  middle.value = display: block; text-align: center;
  small.name = small
  small.value = font-size : 10px;
  large.name = large
  large.value = font-size : 14px;
  gray.name = gray
  gray.value = color: #636563;
  left.name = alignleft
  left.value = float:left; display: block;
}
```

La sélection des couleurs peut aussi être configurée :

```
RTE.colors {
  entreprise {
    name = BT3-Rouge
    value = #BB0000
  }
  variation {
    name = variation Burgundy
    value = #6F0311
  }
}
RTE.default.colors = entreprise, variation
```

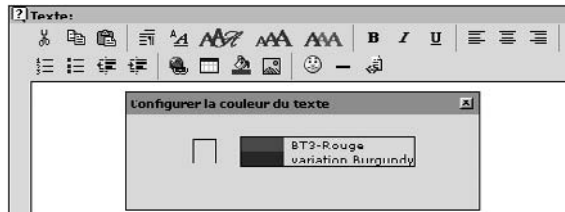


Figure 4.10:
Couleurs
personnalisées dans
le RTE

Afin de restreindre la sélection de couleurs à celles que vous avez vous-même définies, vous pouvez désactiver la palette par défaut comme suit :

```
RTE.default.disableColorPicker=1
```

Voici un exemple d'objet défini par l'utilisateur avec quelques caractères utiles qui ne se trouvent pas sur un clavier :

```
RTE.default.userElements {
10 = Signes Juridiques
10 {
  1 = ®
  1.description = Marque déposée
  1.content = &reg;

  2 = ©
  2.description = Copyright
  2.content = &copy;

  3 = §
  3.description = Paragraphe
  3.content = &sect;
}
20 = Devises
20 {

  1 = ¥
  1.description = Yen
  1.content = &yen;

  2 = £
  2.description = GBP
  2.content = &pound;

  3 = ¢
  3.description = Cent
  3.content = &cent;
}

30 = Symboles mathématiques
30 {
  1 = Puissance
  1.description = exposant
  1.mode = wrap
```

```

1.content = <sup>|</sup>

2 = Indice
2.description = Indice
2.mode = wrap
2.content = <sub>|</sub>

3 = Symbole degré
3.description = symbole degré
3.content = &deg;
}
}

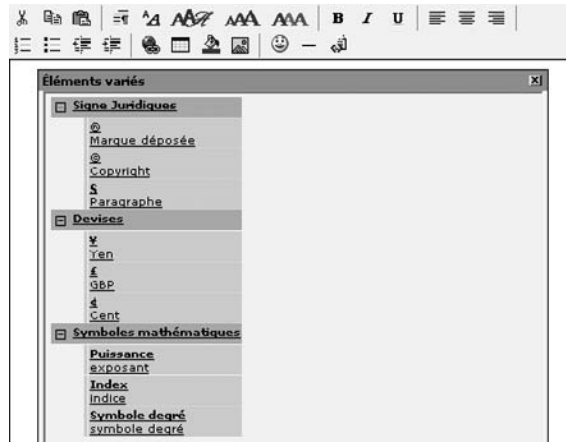
```

Si vous n'avez pas activé tous les éléments dans la barre de menu, n'oubliez pas de rendre l'élément suivant disponible :

```
RTE.default.showButtons = user
```

L'illustration suivante montre le résultat :

Figure 4.11:
Extensions
personnalisées via
l'objet utilisateur du
RTE



Référence 261237 Vous trouverez tous les détails à la référence ci-contre, ainsi que plusieurs autres exemples de configuration du RTE.

Référence 297748 Pour les développeurs, les paramètres des transformations peuvent aussi être insérés à l'aide de TypoScript grâce aux options de la clé **PROC**.

Alternatives

La restriction du RTE à Internet Explorer à partir de la version 5 a entraîné beaucoup d'utilisateurs à envisager d'autres alternatives. Plusieurs essais différents ont eu lieu simultanément afin d'intégrer htmlArea, un autre projet Open Source au sein de TYPO3. Suite à la performance clairement supérieure d'un de ces prototypes, les autres essais sont devenus obsolètes et marqués comme tels dans le répertoire d'extensions du site TYPO3 dont ils doivent (espérons-le) disparaître bientôt. L'extension htmlArea définitive a été conçue par Stanislas Rolland.

L'adresse suivante présente une documentation complète à ce sujet : <http://typo3.org/documentation/document-library/rtehtmlarea/>.

4.8.5 Le module **Web** → **Info** → **Configuration TS de la page**

Maintenant que vos entrées pages sont cachées dans les formulaires d'en-tête de certaines pages, il n'est pas si simple de retrouver quelles valeurs ont été entrées et à quels endroits.

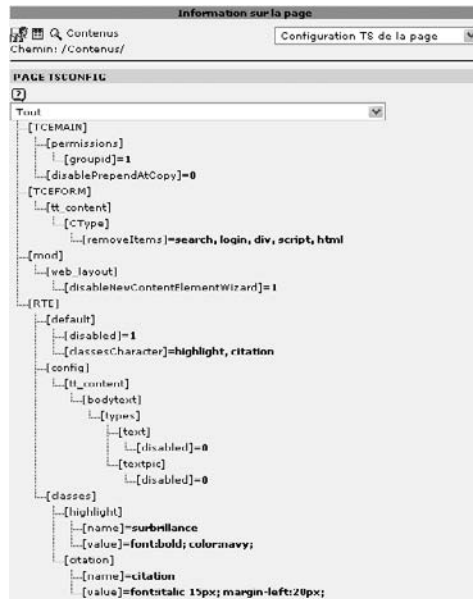


Figure 4.12:
Configuration
TSConfig valables
pour une page prise
au hasard dans
l'arborescence

Le sous-module **Info** résout ce problème grâce au mode **Configuration TS de la page** : il affiche la configuration active de la page sélectionnée. Vous pouvez choisir d'afficher toutes les entrées TSConfig ou seulement certaines sections.

4.9 Créer des Workflows simples

Dans la vue appelée « niveau racine », qui s'affiche dans le module **Liste** lorsque vous cliquez sur le nom correspondant à l'icône en forme de globe terrestre tout en haut de votre arborescence de pages, vous pouvez créer des enregistrements de toute sorte. Si l'extension workflow est installée, vous pouvez maintenant créer un nouveau workflow dans la table **Workflows** en bas de l'affichage liste. Si cette extension ainsi que celles qui en dépendent ne sont pas encore installées, passez d'abord au *gestionnaire d'extensions*. Le chapitre 6 donne l'information nécessaire à l'installation d'extensions.

4.9.1 Configuration d'un workflow

La configuration d'un workflow implique la présence d'une arborescence de pages et d'au moins deux groupes d'utilisateurs backend, avec leurs membres respectifs. Ce module, simplifié pour l'instant, propose trois profils d'acteurs : le gestionnaire de tâches, qui peut assigner des tâches aux utilisateurs, l'auteur, qui crée de nouveaux éléments de contenu, et le rédacteur, qui révisé ce contenu et le publie.

Il y a ici différentes façons de publier :

- Par défaut, les nouvelles pages et les nouveaux éléments de contenu sont invisibles, et demeurent cachés jusqu'à ce qu'ils soient publiés par le rédacteur.
- De nouvelles pages et de nouveaux éléments de contenu sont créés dans une partie privée du site puis, après validation, le rédacteur les déplace vers une partie du site prédéfinie dans le workflow.

Cette façon de procéder facilite la mise en pratique du principe de « double révision » dans les procédures de rédaction. Des workflows plus complexes peuvent être simulés en combinant deux workflows, mais ces options laissent encore à désirer si on considère que le rôle de TYPO3 est d'être au centre des procédures d'entreprise avec les retours d'information que cela implique.

C'est pourquoi un projet a été lancé, impliquant l'intégration de TYPO3 avec un éditeur et un moteur de workflows respectant les standards WfMC. Lors qu'il sera introduit, toutes les options d'affichage seront revues pour créer des workflows plus sophistiqués.

4.9.2 Exemple : workflow d'Actualités

Les utilisateurs du groupe « Portail » devraient être en mesure d'assigner des tâches aux utilisateurs du groupe « B2C », en appelant un workflow prédéfini dans le module **Utilisateur** → **Centre de tâches**. Afin de pouvoir créer un workflow, la page cible, le groupe d'utilisateurs cible ainsi que les extensions nécessaires doivent être disponibles.

Préparation : pour notre exemple, créez une page « Actualités » dans la page « Contenus », et ajoutez-lui deux sous-pages. Nous nommerons « En ligne » la page sur laquelle les actualités doivent être enregistrées pour être ensuite publiées dans le portail et dans les sites, et « Brouillon » la page sur laquelle elles sont créées et éditées en attendant leur publication via le workflow.

Ensuite, à l'aide du module **Web** → **Accès**, accordez les droits suivants aux nouvelles pages :

- « Actualités » : Administrateurs seulement
- « En ligne » : Administrateurs seulement
- « Brouillon » : Groupe « Global », affichage de la page, édition du contenu

Figure 4.13:
Droits d'accès pour
les pages de création
d'actualités

	Propriétaire	Groupe	Tous	Verrouiller
Actualités	*****	***** Global	XXXXX	
En ligne	***** admin	XXXXX	XXXXX	
Brouillon	***** admin	**XXX Global	XXXXX	

Vous devez maintenant ajouter la page « Brouillon » comme point de montage (**DB Mounts**) pour les groupes « Portail » et « B2C » afin que les deux groupes puissent voir et éditer la page. Si vous voulez que le groupe « Portail » soit en mesure d'éditer l'article après sa publication, vous devez lui donner des droits d'accès pour la page « En ligne » et l'ajouter en tant que **DB Mount** (point de montage).



Figure 4.14:
Création d'un
nouveau workflow
dans le niveau racine

Après avoir installé les extensions nécessaires, un nouvel enregistrement de type workflow peut être ajouté dans le niveau racine du système (affichage liste du niveau supérieur de l'arborescence – cf. figure précédente). Le workflow comprend les sections suivantes :

Général

Cette section comprend une case à cocher pour activer/désactiver le workflow et le nom du workflow, ainsi que le champ pour saisir le titre tel qu'il sera affiché pour les utilisateurs. La description saisie ici sera reprise dans la vue détaillée du workflow.

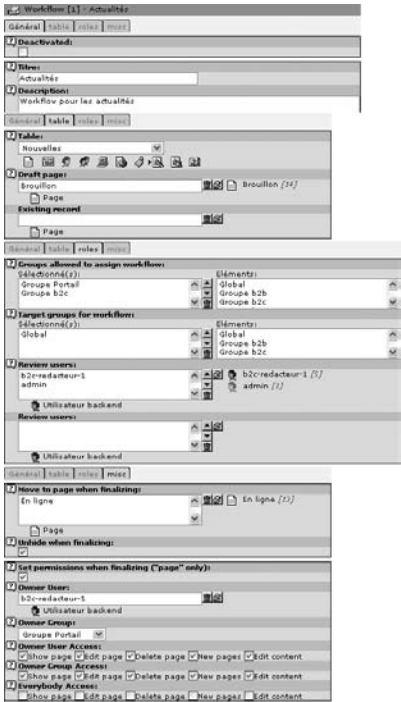


Figure 4.15:
Formulaire de
création du workflow
« Actualités »

Table

Le champ **Table** permet de définir le format cible du workflow. Dans sa configuration d'origine, à condition qu'une extension pour les actualités soit installée, les formats suivants sont disponibles : **Nouvelles**, **Catégorie de nouvelles**, **Page**, **Contenu de la page**, **Membre**, **Groupe de membre**, **Domaine**, **Langue alternative de la page** et **Note interne**. Le champ **Draft page** ouvre le navigateur d'éléments TYPO3, par lequel vous pouvez sélectionner la page dans laquelle les nouveaux éléments seront édités. Sélectionnez ici la page « Brouillon ».

Rôles

Les groupes spécifiés dans **Groups allowed to assign workflow** ont la possibilité de démarrer le workflow et de l'assigner aux groupes repris dans le champ **Target groups for workflow**. Une nouvelle entrée est alors créée dans la liste de tâches des utilisateurs du groupe auquel le workflow est assigné. **Review users** liste les utilisateurs qui peuvent vérifier et publier les résultats. Il est important ici de sélectionner des utilisateurs individuels.

Misc

Vous choisissez ici si l'actualité est déplacée lors de sa publication (**Move to page when finalizing**) par le **Review user** ou si l'actualité passe du statut « caché » à « non caché ».

Si le workflow fait référence à des pages, d'autres champs peuvent être remplis, en cochant la case **Set permissions when finalizing**, afin de spécifier le propriétaire de l'enregistrement — ainsi que le groupe et les droits d'accès pour le propriétaire, le groupe et « tous les autres ».

Après ce champ se trouve une option pour mettre le workflow « Actualités » en action, dans les **Tâches** du module **Utilisateur** → **Centre de tâches**. Il apparaît alors comme une nouvelle tâche « à faire » dans le **Centre de tâches** de l'utilisateur.

Le workflow évolue au fur et à mesure que le statut est modifié. Dans ce processus, chaque étape est enregistrée dans le protocole de tâches et affichée dans l'interface des utilisateurs concernés. Chacun des utilisateurs peut donc faire des ajouts et enregistrer les tâches pour la publication. La publication en tant que telle ne peut être faite que par un utilisateur qui en a l'autorisation.

Par conséquent, dans notre exemple, l'entrée actualités n'est plus marquée en tant qu'objet caché, et a été transférée vers la page « En ligne ». Après que la tâche a été remplie, l'historique des étapes ne demeure visible que pour l'utilisateur qui a initié cette tâche.

Le workflow a été décrit en détail du point de vue du rédacteur à la section 3.4.

4.10 Procédures et actions

Les **actions** sont moins connues, et représentent un concept qui est peu utilisé, mais disponible dans TYPO3 depuis la version 3.3.0. Cette approche met en œuvre des procédures prédéfinies, qui peuvent être complétées par vos propres actions. Les actions sont disponibles dans le module **Utilisateur** → **Centre de tâches** (cf. figure suivante).



Figure 4.16:
Affichage des actions
dans le module
Utilisateur → Centre
de tâches

Cette fonction a été instituée principalement afin de permettre aux administrateurs de créer des accès backend limités. C'est pourquoi d'autres types d'actions standards ont été créées.

4.10.1 Types d'actions

Les fonctions suivantes sont prédéfinies :

Create Backend User

Permet aux non-administrateurs de créer des comptes utilisateurs backend. Pour y parvenir, on doit créer un utilisateur qui peut être copié avec sa configuration — ce qu'on appelle le gabarit utilisateur (*Template user*). Cette fonction est très importante dans des situations où on prévoit un fort accroissement du nombre d'utilisateurs backend dans un groupe utilisateur créé spécialement à cet effet ; l'administrateur évite ainsi les erreurs et un travail fastidieux.

SQL Query

Permet de prédéfinir des requêtes SQL ; pour y parvenir, créez d'abord une nouvelle action dans le niveau racine. Nommez-la et assignez-la à un ou plusieurs groupes. La définition de la requête de base de données est ensuite créée dans le module **Outils** → **Vérification BD** dans le menu **Full Search**, avec l'option **Advanced Query**. Après avoir créé une requête, vous pouvez l'assigner en cliquant sur le bouton **Save**, puis en sélectionnant l'action appropriée (**Save to Action**) dans le menu de sélection. Cette action est maintenant disponible pour tous les membres du groupe spécifié. Rappelez-vous que les utilisateurs doivent avoir le droit d'accès à la table et à la page concernées.

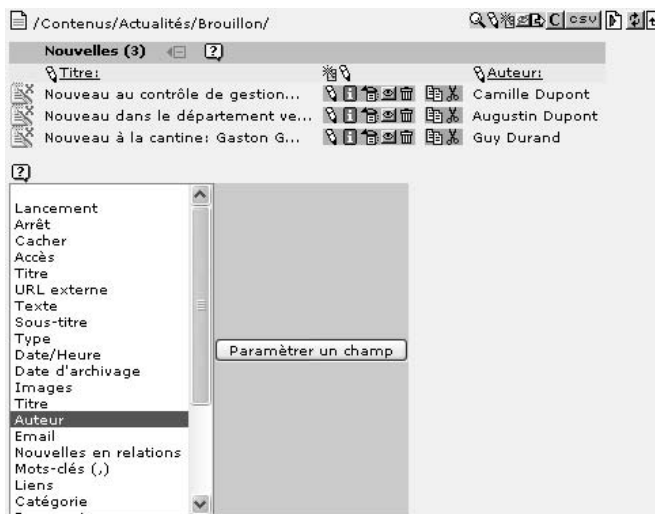


Figure 4.17:
SQL Query : exemple
de requête pour
afficher de
l'information sur les
utilisateurs frontend

Record List

Sert à sélectionner une liste d'enregistrements dans une page spécifique pour être édités et affichés ; l'option apparaît ici comme dans l'affichage liste en mode étendu. Cette option est très utile pour permettre aux utilisateurs d'éditer directement des enregistrements, des produits, des actualités, etc. — après s'être identifiés.

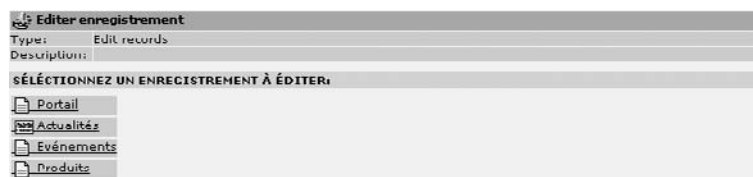
Figure 4.18:
Exemple d'une action
de type Record List



Edit Record

Vous permet d'afficher une liste d'enregistrements dispersés, qu'on peut ensuite appeler pour les éditer. Ainsi, un utilisateur n'étant impliqué que dans une très petite quantité d'enregistrements rarement modifiés pourrait travailler efficacement sans utiliser le module **Web**.

Figure 4.19:
Liste
d'enregistrements à
éditer, appelés par
une action de type
Edit Records



4.10.2 Exemple : action pour créer des utilisateurs

Ce type d'action prédéfinie permet aux non-administrateurs de créer eux aussi des comptes utilisateurs, fonction importante pour obtenir des départements et des équipes autonomes dans une grande société qui, par exemple, s'occuperait de production de contenu.

1. Tout d'abord, l'extension `sys_action` doit être installée via le gestionnaire d'extensions.
2. Afin de pouvoir définir une action pour la création automatique d'utilisateurs, un enregistrement utilisateur doit être disponible en tant qu'utilisateur gabarit (*Template*

User). Cet utilisateur est copié par l'action et reçoit des informations spécifiques telles qu'un nom d'utilisateur, un mot de passe ainsi qu'un point de montage dans l'arborescence. Créez un utilisateur « gabarit_produits ».

Les actions sont créées dans le module **Liste** du niveau racine (les administrateurs peuvent aussi les appeler directement à-partir du centre de tâches). En cliquant sur **Créer un nouvel enregistrement**, le dialogue pour un nouvel enregistrement s'affiche. Sélectionnez le type **Action**.

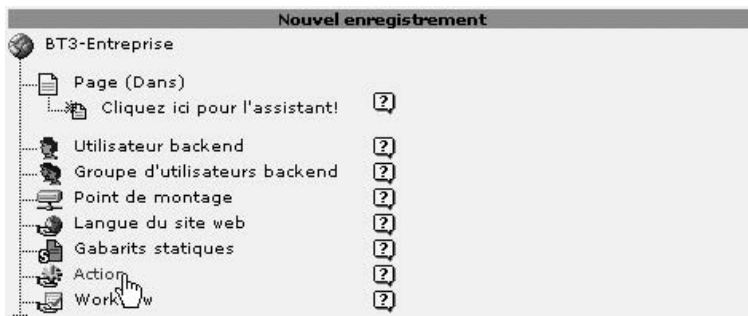


Figure 4.20:
Création d'une
nouvelle action dans
le niveau racine

Dans le formulaire de création, sélectionnez l'option **Create Backend User** et remplissez les autres champs (cf. figure suivante) :

Titre/Description

Saisissez un titre et une description

Assign Action to groups

Sélectionnez les groupes qui seront autorisés à utiliser cette action via le centre de tâches. Sélectionnez ici le groupe « Portail ».

User prefix

Vous permet d'entrer une expression qui précédera le nom de tous les utilisateurs créés à l'aide de cette action. Dans notre exemple, entrez « pm » pour « product manager ». Ces utilisateurs sont ensuite regroupés dans la liste des utilisateurs backend et donc facilement reconnaissables.

Template user

Choisissez l'utilisateur du niveau racine qui, à cause de ses paramètres de base (y compris les champs de type **Allowed Exclude Fields**), servira de gabarit et sera copié. Sélectionnez l'utilisateur « gabarit_produits » créé plus tôt.

Figure 4.21:
Formulaire d'action
pour la création d'un
gestionnaire produit

The screenshot shows the 'Action [4] - Configurer nouveau gestionnaire produit' form. It includes fields for 'Deactivated' (checkbox), 'Type' (set to 'Create Backend User'), 'Titre' (set to 'Configurer nouveau gestionnaire produit'), and 'Description' (a text area with instructions). Below these are two list boxes for 'Assign action to groups': 'Sélectionné(s)' (containing 'Groupe b2b' and 'Groupe b2c') and 'Éléments' (containing 'Global', 'Groupe b2b', 'Groupe b2c', 'Groupe b2e', and 'Groupe Portail'). Other sections include 'User prefix' (set to 'pm'), 'Template users' (with a dropdown and a link to 'gabarit-produits /8/'), 'Groups which may be assigned through the action' (with 'global' selected), and 'Create User Home Directory' (checkbox).

Groups which may be assigned through the action

On peut aussi choisir à quel groupe ce nouvel utilisateur sera affilié dans la liste proposée. Choisissez le groupe « Global ».

Create User Home Directory

Dans le champ **All configuration** de l'outil d'installation, vous pouvez entrer les deux valeurs suivantes :

```
$TYPO3_CONF_VARS[ "BE" ] [ "userHomePath" ]  
$TYPO3_CONF_VARS[ "BE" ] [ "lockRootPath" ]
```

On doit bien sûr avoir les droits d'écriture sur le répertoire spécifié.

L'action est créée lorsque vous l'enregistrez. Les utilisateurs du groupe correspondant peuvent maintenant la trouver dans leur module **Utilisateur** → **Centre de tâches**. En tant qu'administrateur, vous pouvez aussi appeler l'action à partir de ce module pour vérifier les champs d'entrée visibles d'une part et tester la fonction d'autre part.

Figure 4.22:
Formulaire de
création d'utilisateur
dans le module
Utilisateur → Centre
de tâches

The screenshot shows the 'Configurer nouveau gestionnaire produit' form within the 'UTILISATEURS BACKEND' section. It lists existing users like 'pm_proche1 (camille dupont)' and 'pm_proche2 (augustin dupont)'. Below is the 'MODIFIER UNE UTILISATEUR BACKEND' section for 'pm_proche1 (camille dupont)'. It includes checkboxes for 'Désactiver l'utilisateur' and 'Utilisateur', and input fields for 'Nom d'identification' (set to 'pm_proche1'), 'Mot de passe', 'Nom' (set to 'camille dupont'), 'E-mail' (set to 'camille@dupont.fr'), 'Membre de(s) groupe(s)' (with 'global' selected), and 'Départs de l'arborescence' (set to 'Produit 2'). At the bottom are 'Mettre à jour' and 'Effacer' buttons.

4.11 Administration des utilisateurs frontend

Les utilisateurs frontend (« FEUtilisateur ») sont des visiteurs d'une page Internet qui obtiennent davantage de droits, en s'enregistrant ou en se faisant créer un compte par l'administrateur, afin d'accéder à des éléments de contenu protégés ou d'utiliser des fonctions qui ne sont pas disponibles pour des utilisateurs non-enregistrés.

Les utilisateurs frontend peuvent être repris dans des groupes qui se voient ensuite assigner les dites permissions au niveau d'une page, ou pour des éléments de contenu spécifiques.

Le système d'utilisation frontend fournit une base pour accorder les droits d'accès et, si nécessaire, les droits d'édition aux visiteurs du site Web qui se sont identifiés correctement. TYPO3 propose une série de plugins frontend afin de rendre ce type d'édition possible. Un livre d'or ou une carte de vœux virtuels sont des exemples simples d'utilisation du frontend. La liste des extensions disponibles est tenue à jour dans le gestionnaire d'extensions. Dans le backend, les utilisateurs frontend et leur groupe sont créés dans une page de type Dossier Système.

4.11.1 Création de groupes d'utilisateurs

Pour des besoins de clarté, les comptes utilisateurs et les groupes d'utilisateurs sont enregistrés dans une section différente de l'arborescence. Créez une nouvelle page à cet effet. Sélectionnez le type de page **Dossier Système** et nommez-la « FEUtilisateur ». En sélectionnant l'extension **Membres** dans le menu **Contient le plugin**, vous assignez une icône appropriée au répertoire.



Figure 4.23:
Groupes et
utilisateurs frontend

Retournez maintenant au module **Page** en cliquant sur **Enregistrer et fermer le document**. Vous pouvez ajouter un nouveau groupe d'utilisateurs en cliquant sur **Créer un nouveau contenu**. Créez le groupe « B2B » et confirmez l'enregistrement en cliquant sur **Enregistrer et fermer le document**. Répétez le processus, créez un nouveau répertoire dans la section de l'arborescence « B2E » et ensuite, à l'intérieur de ce répertoire, le groupe « B2E ».

Vous pouvez maintenant voir tous les groupes et utilisateurs dans l'affichage liste.

4.11.2 Création de comptes utilisateurs

La prochaine étape consiste à ajouter des utilisateurs individuels au groupe d'utilisateurs. Afin de remplir les différentes tâches dans notre exemple, nous créerons un utilisateur « entreprise_1 » et « employe_1 » dans leurs groupes et répertoires respectifs. Complétez un maximum de champs lors de leur création, afin de découvrir la validation automatique de certains types de champ, et de pouvoir par la suite tester l'interaction avec d'autres applications – telles que le système DMAIL – avec ces éléments de contenu de la base de données.

4.11.3 Identification

Vous devez prévoir quelque part dans votre site un endroit où les utilisateurs frontend pourront s'identifier. Insérez un élément de contenu de type **Identifiant** dans une page. Compte tenu du fait que TYPO3 est capable de gérer plusieurs sites dans un même système, et que vous voudrez probablement garder les utilisateurs Internet des différents sites dans des sections séparées, le système doit savoir à quel enregistrement le formulaire d'identification fait référence. C'est pourquoi vous devez saisir l'expression suivante dans les constantes du gabarit de la page où se situe l'identifiant :

```
styles.content.loginform.pid = 50
```

« 50 » fait ici référence à l'UID du **Dossier Système** dans lequel vous avez créé les utilisateurs et les groupes un peu plus tôt. Vous trouvez cet UID en cliquant sur l'icône Info qu'on retrouve dans le module **Liste**. L'UID de cet enregistrement apparaît à la dernière ligne au bas de la fenêtre d'information.

Figure 4.24:
Information dans le
module Liste



Il est aussi possible (et plus rapide) d'afficher l'UID en déplaçant votre curseur sur l'icône du dossier dans l'arborescence.

Figure 4.25:
Affichage de l'UID en
plaçant la souris sur
l'icône



4.11.4 Assigner des pages et des éléments de contenu

Afin d'assigner du contenu dont vous voulez réserver l'accès à ce groupe, afficher le formulaire d'en-tête de la page désirée ou le formulaire d'édition de n'importe quel élément de contenu. Dans la dernière section du formulaire se trouve l'option **Accès**. Vous pouvez sélectionner un des groupes d'utilisateurs que vous avez créés précédemment. L'option **Cache à la connexion** cache automatiquement le contenu de la page après que l'utilisateur s'est identifié. **Afficher à toutes les connexions** affiche le contenu lorsque l'utilisateur s'est identifié, peu importe son groupe d'appartenance. L'option **Inclure les sous-pages** étend la protection de l'accès à toute la section de l'arborescence se rattachant à la page, et l'icône de la page comporte alors deux petites flèches.

Exemple

Nous exécutons à présent les deux étapes suivantes de notre exemple :

1. Insérez des éléments de contenu **Identifiant** dans les pages « Accueil B2B » et « Accueil B2E ».
2. Assignez les deux pages aux groupes d'utilisateurs frontend correspondants et activez l'option **Inclure les sous-pages**.

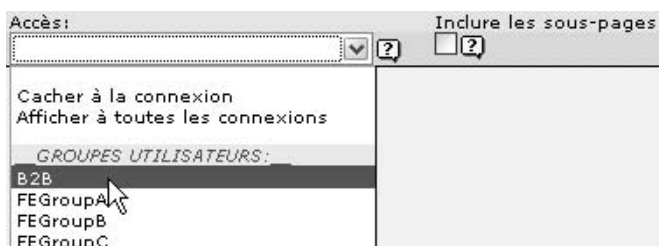


Figure 4.26:
Assignation d'un
enregistrement à un
groupe d'utilisateurs
frontend

3. Ajoutez la ligne suivante dans la section **Setup** du gabarit des pages « Accueil B2B » et « Accueil B2E » :

```
styles.content.loginform.pid = uid
```

"uid" doit être remplacé par les **UID** respectifs des pages FEUtilisateur dans lesquelles sont enregistrés les groupes d'utilisateurs.

4.11.5 Perspectives

Des extensions ont récemment apporté plusieurs améliorations au système des utilisateurs frontend. Il est désormais possible — et il s'agit d'un changement très attendu — d'assigner plusieurs pages et éléments de contenu à différents groupes d'utilisateurs en même temps. Une extension correspondante du noyau de TYPO3 est aussi en cours de développement, ainsi que la possibilité d'afficher la hiérarchie des groupes dans la sélection.

Le système d'utilisateurs frontend fournit principalement la base pour la personnalisation et l'édition des fonctionnalités de portail. Son développement se poursuit, poussé par des attentes sans cesse grandissantes. C'est pourquoi il est recommandé, suivant les fonctionnalités prévues dans votre site, de vérifier régulièrement quelles sont les nouvelles possibilités offertes par les extensions.

4.12 Statistiques et logs

Grâce à une série de paramètres du système, TYPO3 peut afficher des statistiques d'utilisation du logiciel à tous les niveaux : allant du travail des rédacteurs aux statistiques des visiteurs du site, en passant par le type de navigateur utilisé, jusqu'à supprimer l'information sur les enregistrements de données dans TYPO3.

Les notions des statistiques à propos des visiteurs — telles que le nombre de visites par page, les intervalles de temps entre les visites ou les données système du visiteur — étant généralement suffisamment familières, nous n'abordons ici que les données propres à TYPO3. Dès que vous

Référence 471055

êtes confronté à l'analyse des performances de votre site Web, une série d'outils sont à votre disposition. Une liste de liens (vers des documents en anglais) proposés par le « Centre for Information Quality Management » sont disponibles (voir la référence ci-contre).

4.12.1 Le module Web → Info

Ce module comporte quelques options d'analyse intéressantes :

Arborescence (vue d'ensemble)

Trois options sont disponibles dans cette vue d'ensemble : **Paramètres de base**, **Cache et âge**, et **Enregistrement (vue d'ensemble)**. Tous les paramètres concernant les pages de l'arborescence s'y trouvent et peuvent être édités directement.

Localisation (vue d'ensemble)

La vue d'ensemble de la localisation est relativement nouvelle et propose une façon simple et rapide de voir quelles pages ont été récemment modifiées dans la langue par défaut, et de vérifier si leur traduction existe dans les autres langues.

Log

Affiche la même information que le module **Outils → Fichier journal**, mais seulement pour la partie de l'arborescence sélectionnée ; vous constaterez cette différence en lisant plus loin la description du module.

Configuration TS de la page Comme nous l'avons déjà décrit à la section 4.8, cette option affiche les entrées valides dans les champs **TSConfig** de la page sélectionnée.

Statistiques d'affichage

Dans un souci d'exhaustivité, nous devons mentionner ici que TYPO3 possède dans le module **Info** une fonction rudimentaire qui affiche les accès aux pages. Son développement a été freiné par l'arrivée d'un outil de statistiques externe, **AWStats**. Nous recommandons fortement l'utilisation de ce dernier, qui demeure inégalé pour l'analyse des fichiers log du serveur dans TYPO3.

4.12.2 Intégration d'AWStats

Le module AWStats, développé par notre auteur René Fritz, propose une analyse des statistiques d'un site Web.

AWStats est une extension qui doit d'abord être installée dans le système – comme décrit dans le chapitre sur le gestionnaire d'extensions. Puisque l'extension intègre le logiciel AWStats, écrit en Perl, Perl doit être installé sur le serveur.

Dans le fichier **localconf.php**, spécifiez où TYPO3 devra écrire ses fichiers log devant être analysés par AWStats. Utilisez à cet effet le champ de l'**Install Tool** (dans le module **Installation**) sous **All Configuration** :



[logfile_dir]
 Path where TYPO3 should write webserver-style logfiles to. This path must be write-enabled for the webserver. If this path is outside of PATH_site, you have to allow it using [BE][lockRootPath]

[FE][logfile_dir] = fileadmin/
 fileadmin/

Figure 4.27:
 Spécification du
 chemin pour
 l'enregistrement des
 fichiers log afin d'être
 analysés par AWStats

Vous pouvez spécifier n'importe quel répertoire, sans oublier que TYPO3 doit en avoir les droits d'écriture. Les fichiers log générés par TYPO3 sont écrits en format Apache et peuvent aussi être analysés par d'autres outils qu'AWStats.

Le code TypoScript suivant doit maintenant être entré dans le gabarit de base du site Internet, afin d'activer la génération de fichiers log :

```
config.stat = 1
config.stat_apache = 1
config.stat_apache_logfile = domaine.txt
```

Vous pouvez configurer un autre fichier log à n'importe quel endroit de l'arborescence avec `config.stat_apache_logfile`, afin de créer des statistiques indépendantes pour des sections spécifiques. Il est logique de créer un fichier log pour chaque site ou domaine au sein d'une installation TYPO3.

L'expression :

```
config.stat_mysql = 0
```

permet de désactiver l'option de TYPO3 qui enregistre les données log dans sa propre table de statistiques. Le fichier `domaine.txt` doit maintenant être créé dans le répertoire spécifié. Si vous avez choisi le répertoire `fileadmin/`, comme nous le faisons dans l'exemple, vous pouvez créer un fichier correspondant dans le module **Fichier** → **Fichiers** à l'aide de la commande **Nouveau**.

Pour finir, vous devez ouvrir et configurer le module AWStats dans le backend. Après avoir cliqué sur **Éditer la configuration** vous verrez les fichiers `.txt` et `.log` qui ont été trouvés dans le répertoire spécifié. Entrez dans le champ correspondant les domaines auxquels ils doivent s'appliquer. Si vous avez plusieurs domaines, séparez-les par une virgule.

Après avoir enregistré la configuration, vous devez cliquer sur le fichier log pour afficher les statistiques. Ce module ne s'actualise pas de lui-même : vous devez cliquer sur **Mise à jour immédiate** afin d'obtenir les dernières statistiques.

Après avoir bien installé et configuré AWStats, ce dernier affichera une image claire de tous les paramètres importants d'analyse du trafic, tels que les impressions de pages et le nombre de *visiteurs différents*, qui peuvent être classés par domaine. La référence ci-contre renvoie au site d'AWStats où se trouve un glossaire des termes utilisés.

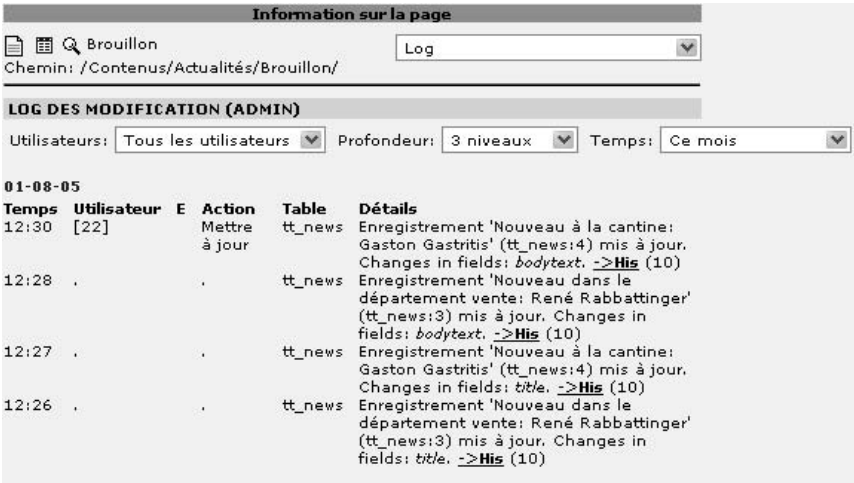
Référence 866808

4.12.3 Analyse des fichiers journaux

Le module **Fichier journal** fournit une vaste gamme d'informations sur les opérations effectuées dans le backend. Avec les paramètres **Users**, **Max**, **Time** et **Action**, vous procédez à

différents types d'analyse de la base de données. L'information la plus importante est fournie par l'heure (Time) et le nom d'utilisateur (Login), qui peuvent être complétés par les opérations effectuées par l'utilisateur, si vous sélectionnez **All** dans le menu **Action**. En plus du nom, de l'heure, de l'adresse IP et de l'ID de l'enregistrement édité, vous pouvez aussi afficher l'historique d'édition à l'aide du lien **His** en caractère gras.

Figure 4.28:
La fonction log par le module Web → Info pour une section de l'arborescence



4.12.4 Logs frontend

Deux autres extensions sont disponibles, qui fournissent de l'information sur les visiteurs et les utilisateurs frontend du système :

Référence 999758 Le Visitor Tracking System de Carlos Chiari (clé d'extension : `de_phpote`) enregistre le chemin suivi par les visiteurs dans le site Web et affiche soit le chemin suivi, soit des statistiques.

Référence 779472 Le Login User Tracking (clé d'extension : `loginusertrack`) de Kasper Skårhøj enregistre les données relatives aux sessions des utilisateurs frontend.

Vous trouverez de l'information sur ces deux extensions aux références mentionnées ci-contre.

4.12.5 Le module Vérification BD

La vérification de la base de données effectue une analyse de cohérence, et peut devenir un outil important pour les systèmes complexes supportant une lourde charge, en relevant les problèmes et en les aidant à les résoudre.

Le module propose les fonctions suivantes :

Record Statistics

Cette option analyse toutes les entrées dans les tables de la base de données qui sont utiles à l'administrateur. **Marked-deleted pages** : la première section affiche des données de base sur le nombre de pages qui sont cachées, qui ne le sont pas, ou qui sont supprimées. Cette fonctionnalité est particulièrement utile parce que les pages supprimées du backend ne sont pas effacées de la base de données ; le champ `deleted` de

la table **pages** est simplement mis à la valeur « 1 ». Ainsi s'il s'avérait nécessaire en cas d'urgence de récupérer une page effacée, cette valeur pourrait être ramenée à « 0 » via l'outil de base de données *phpMyAdmin*, afin de rendre la page à nouveau visible.

Document Types affiche la fréquence des différents types de page. La section **Tables** fournit de l'information sur le nombre d'entrées existantes, et affiche le nom de la table du système dans la colonne du milieu.

Total Page Tree

Cette option est particulièrement intéressante pour des installations de très grandes dimensions, afin de visualiser l'arborescence dans son ensemble sans avoir à ouvrir ou fermer plusieurs sections.

Database Relations

Cette section effectue une analyse de la base de données selon des critères spécifiques.

Files with no references at all

Le système effectue un balayage des fichiers contenus dans le répertoire **Uploads** dans lequel sont enregistrés tous les fichiers chargés via le backend sur le serveur sans recevoir de destination spécifique. Ces fichiers sont ensuite analysés pour voir s'ils sont référencés par la base de données. Si ce n'est pas le cas, ces fichiers peuvent, et devraient être effacés, puisqu'ils ne sont plus utilisés et ne peuvent être appelés par les utilisateurs du backend. De plus, ils peuvent occuper un espace important sur le disque dur de grandes installations ; en effet, certains utilisateurs enregistrent des images de cette façon sur le serveur.

Files referenced from more than one record

Les entrées de la base de données qui sont copiées peuvent présenter plusieurs références vers un fichier du répertoire **Uploads**. Ces liens multiples sont analysés et affichés sous cette rubrique.

Missing Files

Cette option recherche à travers la base de données des liens vers des fichiers du répertoire **Uploads** qui n'existent plus.

Select Fields/Group Fields

Cette fonction recherche, dans les enregistrements de la base de données, des liens vers d'autres enregistrements n'existant plus. *Select fields* et *Group fields* font référence à différentes façons d'afficher des affectations de ce genre dans TYPO3. *Select fields* est utilisé pour des affectations à partir d'une sélection provenant d'une table dans la base de données (ex. : groupes dans le formulaire de définition d'un utilisateur), et *Group fields* est utilisé pour des affectations qui se réfèrent à du contenu provenant de différentes tables de la base de données.

Full Search

Permet d'effectuer des recherches dans la base de données sans avoir à passer par *phpMyAdmin*.

Raw Search in all fields

En mode simple, vous pouvez effectuer des recherches dans toute la base de données en entrant une expression et en la soumettant. Cette action n'est pas

recommandée pour de très grandes bases de données ou des systèmes travaillant sous de très grandes charges.

Advanced query

Offre une série d'options intéressantes. L'affichage des résultats peut être modifié à l'aide des options suivantes :

Select Records affiche les résultats en une liste pouvant être éditée directement.

Count results se contente d'afficher le nombre d'occurrences.

Explain Query montre les paramètres plus avancés de la requête.

CSV Export affiche le résultat en une liste dont les éléments sont séparés par des virgules. Chaque ligne contient un enregistrement avec les valeurs affichées entre guillemets. Ce format est particulièrement utile si vous voulez traiter ces données dans des tableurs, des programmes pour faire des statistiques, etc. Le résultat apparaît dans une fenêtre au bas de la page, avec une option pour le télécharger.

XML Export affiche le résultat en format XML dans une fenêtre, ici aussi avec une option pour le télécharger.

Make Query : après avoir sélectionné la table de base de données dans laquelle vous désirez effectuer une recherche, vous pouvez spécifier des champs qui seront ensuite ajoutés à la liste de champs devant être vérifiés. Vous pouvez spécifier les clauses de condition de la recherche en dessous. Le nom de la table s'affiche à gauche, l'opérateur au centre, et l'expression recherchée à droite.

Les opérateurs possibles sont **contains**, **starts with**, **ends with** et **equals**. La case à cocher à la droite du champ inverse les opérateurs : **does not contain**, **does not start with**, **does not end with**, **does not equal**.

Chaque ligne comporte les options **Update**, **Remove condition** et **Add condition**. Chaque nouvelle ligne offre aussi la possibilité de changer l'ordre ou de modifier les conditions. La flèche vers la droite pour l'indentation vous permet de définir des conditions avec les opérateurs « and/or ». Sous les lignes pour la définition des requêtes se trouvent trois fonctions pour grouper le résultat. **Group by** permet d'agréger le résultat en fonction d'un champ de la table, **Order by** affiche les lignes du résultat classés selon un champ, et **descending** affiche le résultat en ordre décroissant. **Limit** permet de déterminer le nombre maximum de lignes dans le résultat.

Find Filename

Offre une option simple pour effectuer des recherches par nom dans tous les fichiers du point de montage et de la source TYPO3.

Référence 592203

On peut effectuer une recherche avec des expressions régulières; la référence ci-contre en donne une courte présentation, ainsi que des liens vers d'autres ressources.

Relations:

Find filename

PATTERN

Enter regex pattern:

SEARCHING FOR FILENAMES:

fileadmin/ being checked...
 Dirs: 34
 Files: 80
 Matching files:

media/ not checked.

tslib/ not checked.

typo3/ not checked.

typo3conf/ not checked.

typo3temp/ being checked...
 Dirs: 5
 Files: 465
 Matching files:

uploads/ being checked...
 Dirs: 6
 Files: 74
 Matching files:

typo3_src/ being checked...
 Dirs: 275
 Files: 5306
 Matching files:
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom1_bottom.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom1_mask.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom2_bottom.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom2_mask.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom3_bottom.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom3_mask.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom4_bottom.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom4_mask.jpg
 typo3_src/typo3/sysex/t/cms/tslib/media/frames/darkroom5_bottom.jpg

Figure 4.29:
Résultat de la
recherche de fichiers

4.13 TYPO3 et le système de cache

En tant que CMS s'appuyant sur une base de données, TYPO3 génère des pages HTML à partir de plusieurs sources différentes. Depuis des gabarits HTML, via TypoScript en provenance de la base de données, ou de contenu de page provenant d'applications différentes. Les pages sont assemblées lorsqu'elles sont appelées et transférées vers le navigateur du visiteur. Parce que les fonctions des scripts PHP ainsi que les requêtes nécessaires représentent une lourde charge pour le serveur Web, TYPO3 enregistre le résultat des pages qui ont été assemblées pour la première fois dans une mémoire tampon, qu'on appelle le *cache*. Ce cache diminue considérablement la charge sur la base de données et sur le serveur, surtout pour de grands sites Web, soumis à un trafic très élevé.

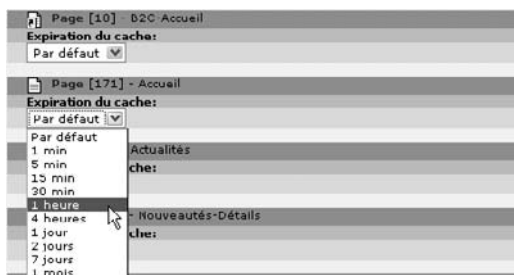
Lors de modifications, le cache de la page en question n'est plus à jour. Plusieurs fonctions sont donc fournies à l'administrateur afin d'effacer ce cache manuellement.

En dessous de la liste des modules, vous trouvez deux fonctions globales pour effacer, d'une part le cache des pages, d'autre part le cache des fichiers de configuration.

Le module **Web** → **Page** propose plus d'options dans le bas de la vue détaillée. Le cache d'une page peut y être vidé manuellement, ainsi que celui d'un nombre déterminé de sous-pages.

Le mode **Cache et âge** de l'aperçu **Arborescence (vue d'ensemble)** dans le module **Web** → **Info** sert d'outil de contrôle. Vous pouvez y vérifier non seulement les caches actuels des pages, mais vous pouvez aussi modifier la durée de leur validité en cliquant sur l'icône « crayon » dans le haut de la colonne **Cache**.

Figure 4.30:
Réglages du cache
d'une page via le
module Web → Info



4.14 Digital Asset Management

4.14.1 Tâches et buts du DAM

Le *Digital Asset Management System* (DAM) est la réponse de la communauté TYPO3 au besoin de gérer les images, documents et autres fichiers (ce qu'on appelle les ressources)⁸ et leurs méta-données afin d'enrichir le site Web. Dans ce contexte, un système de gestion de ressources digitales est un sous-système n'ayant pas de fonction directement visible dans le site Web ; il sert à enregistrer, indexer et gérer les fichiers. Toutefois, il fournit aussi une interface simple pour les extensions qui utilisent cette information pour exécuter certaines fonctions dans le backend ou le frontend.

Une autre innovation importante est qu'il résout un point faible de TYPO3, souvent critiqué : comment manipuler les fichiers lorsqu'ils sont référencés à partir d'éléments de contenu ? TYPO3 copie les fichiers devant être publiés dans une partie séparée de l'arborescence de fichiers du système afin d'empêcher l'utilisateur d'accéder aux originaux. Si un utilisateur efface un fichier du système à partir du backend, la fonction de contenu affichant ces fichiers demeure inchangée, et est protégée contre des erreurs de la part de l'utilisateur. Dans plusieurs cas toutefois, il peut être souhaitable d'avoir un accès direct afin, par exemple, de remplacer un fichier qui est utilisé en plusieurs endroits du site. Mais la fonction de protection empêche ce type d'action.

⁸NdT. : appelés *assets* en anglais

Le DAM propose une façon de configurer au cas par cas la manipulation de fichiers, afin que vous puissiez spécifier par fichier, par groupe ou par catégorie si le fichier doit être copié vers un des répertoires temporaires ou s'il doit rester là où il est et être utilisé directement à partir de sa source.

Un exemple simple d'application du DAM serait une galerie reprenant des images, indépendamment de l'endroit où elles ont été enregistrées, mais sur base des méta-données telles qu'une catégorie spéciale créée à cet effet, ou un mot-clé, ou encore un type de fichier.

On pourrait aussi l'utiliser pour l'analyse statistique de l'utilisation d'images provenant d'une réserve de matériel à la carte, et fournir une interface qui en indiquerait les résultats dans le backend.

4.14.2 Intégration dans TYPO3

Dans le backend, le DAM prend la forme d'un nouveau module principal et est aussi intégré dans l'arborescence des pages. L'icône DAM ouvre toujours une vue de toutes les catégories ainsi qu'une liste des types de fichiers. Le sous-module **Media** → **List** montre tous les fichiers sous forme de liste. En pratique les modules **Media** → **List** et **Media** → **File** devraient remplacer le module standard **Fichier** → **Fichiers**. En fonction de la configuration, le module DAM est soit affiché avec toutes ces vues dans le backend rédacteur, soit se limite à l'intégration DAM dans le navigateur d'éléments TYPO3. Dans ce cas, la gestion des données image est réservée à l'administrateur.

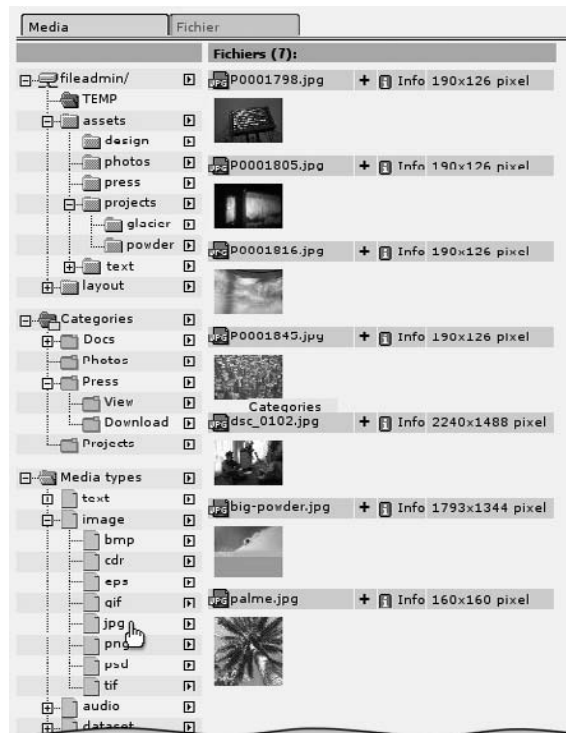
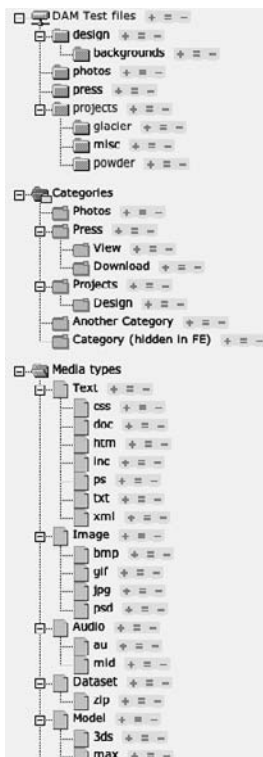


Figure 4.31:
Le navigateur
d'éléments TYPO3
lorsque le DAM est
installé

Le module Media → List

La liste vous permet de travailler avec et sur les méta-données de fichiers déjà contenus dans le DAM, et ses fonctions sont compatibles avec le module standard **Fichier** → **Liste**. Les rédacteurs peuvent l'utiliser pour modifier des méta-données, faire des sélections et les enregistrer. Un système simple d'indexation permet d'éditer massivement les méta-données de ressources déjà existantes.

Figure 4.32:
Arborescence
symbolique du DAM
avec trois sections
physiques : la
structure physique du
répertoire, les
catégories et les
types de médias



Comme on peut le voir dans la figure précédente, il existe trois arborescences différentes dans l'aire de navigation. La première reprend les fichiers du point de montage, la seconde montre la hiérarchie des catégories de contenu. La dernière est automatiquement générée à partir du système d'indexation. Les objets multimédias y sont listés et regroupés par type de fichier.

En cliquant sur le nom d'une section de l'arborescence, une liste des médias indexés pour ce répertoire, catégorie ou type de fichier s'affiche dans la vue détaillée. Vous pouvez y éditer les données des objets médias en cliquant sur l'icône « crayon » adjacent à l'objet en question.

Hormis l'affichage liste, il existe trois autres modes proposant leurs propres fonctions après que vous avez sélectionné un répertoire, une catégorie ou un type de média :

Vignettes

Les images, s'il y en a, sont affichées sous forme de vignettes.

Sélection

On peut y modifier la sélection, par exemple en retirant certains fichiers de la sélection. De plus, il est possible d'enregistrer la sélection et de l'échanger entre rédacteurs.

Process

On peut y traiter les fichiers en modifiant ou en ajoutant des méta-données.

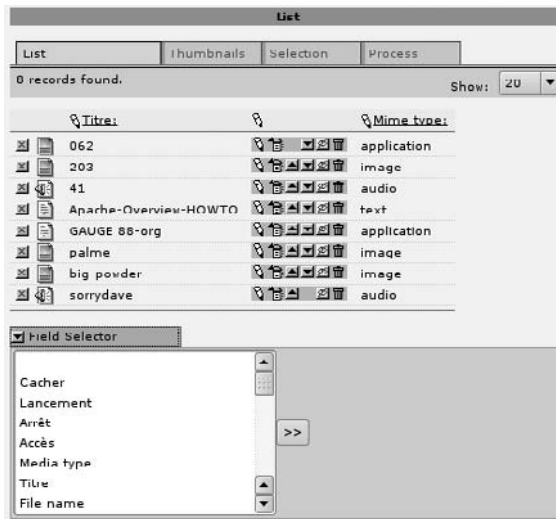


Figure 4.33:
Affichage liste dans
le DAM

Le module Indexing

Le module **Indexing** sert d'interface d'importation afin de fournir de la méta-information aux nouveaux fichiers et de les ajouter au système. Le module DAM crée un objet média pour chaque fichier inséré dans la base de données, qui contient la méta-information correspondante. Pendant l'indexation, le module, en fonction du type de fichier, lit automatiquement différentes informations par défaut contenues dans la base de données (ex. : dimensions de l'image, taille du fichier, etc.). La routine d'indexation est prévue pour permettre l'ajout de ce qu'on appelle des *services*, qui pourraient éventuellement lire de la méta-information à partir de fichiers PDF ou Office par exemple.

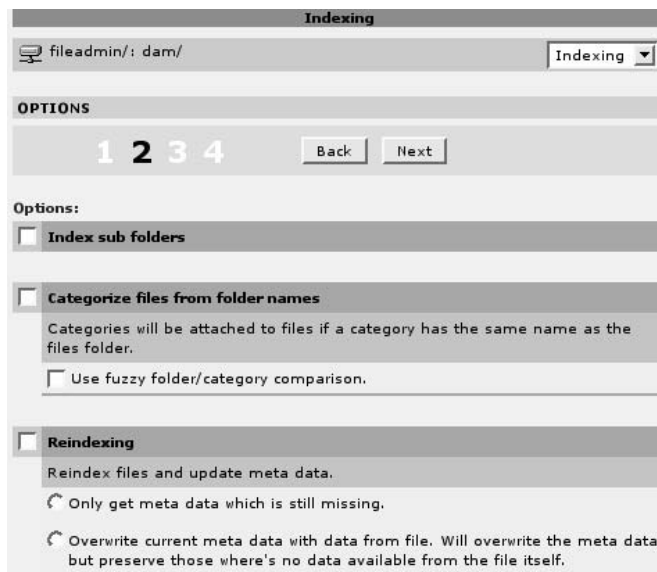
L'indexation de méta-données requiert les étapes suivantes :

1. Téléchargez les fichiers vers un sous-répertoire de fileadmin via le module **Fichier** → **Liste** (ou via FTP).
2. Importez via **Media** → **Indexing** en sélectionnant le répertoire qui doit être indexé dans l'arborescence fileadmin.
3. Suivez les instructions de l'assistant d'indexation, comme le montrent les illustrations suivantes :

Figure 4.34:
Étape 1 : sélectionnez
le répertoire dont
vous voulez insérer le
contenu dans le DAM



Figure 4.35:
Étape 2 : sélectionnez
des options de la liste



Indexing

fileadmin/: dam/ Indexing

INDEXING FIELD PREDEFINITION

1 2 3 4 Back Next

Preset meta descriptions for files.
The content may be replaced by meta content from the files themselves.

▼ Check the fields you want to set **fixed**.
The input will then not be replaced.

Title:

☐

Keywords (,):

☐

Description:

☐

Source/Original location:

☐

Source/Original location description:

☐

Ident key (sku):

☐

Indexing

fileadmin/: dam/ Indexing

INDEXING SETUP SUMMARY

1 2 3 4 Back Start

Set Options:

Index sub folders

Categorize files from folder names

Reindexing

• Files will be reindexed. Only meta data will be get which is still missing.

Meta data preset:

Title:

Keywords (,):

press

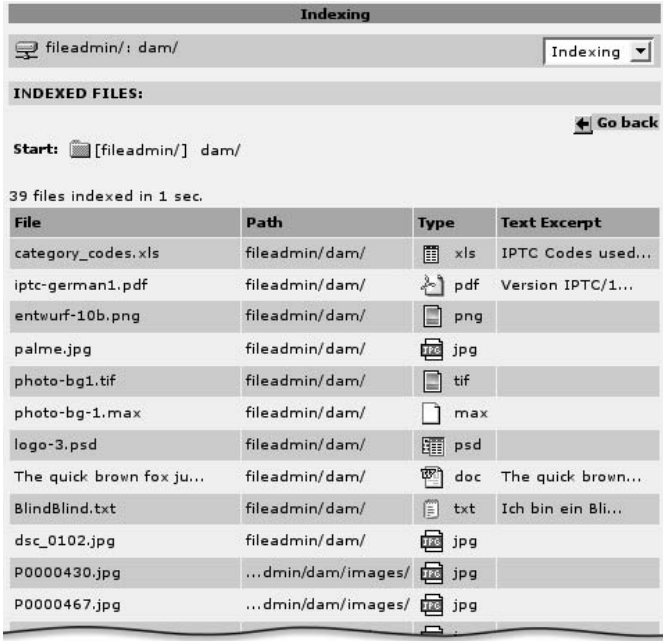
Description:

Figure 4.36:
Étape 3 : entrez
l'information que
vous voulez assigner
aux fichiers en vue
d'un traitement en
masse

Figure 4.37:
Étape 4 : résumé de
vos entrées ; elles
peuvent être
enregistrées en tant
que modèles pour
d'autres indexations

189

Figure 4.38:
Étape 5 : affichage
des fichiers traités



Catégories DAM

Le module DAM crée une page spécifique dans l'arborescence de pages qui sert de répertoire où il place ses propres enregistrements (objets médias et catégories). Les catégories peuvent être créées, éditées et effacées via le module **Web** → **Liste**. Tous les objets médias DAM se retrouvent aussi dans ce répertoire, mais sont édités de façon plus efficace dans le module **Media** → **List**.

4.14.3 Perspectives

Le DAM comble un manque en ce qui concerne la manipulation d'informations et de ressources basées sur des fichiers plutôt que sur des enregistrements de la base de données, et deviendra un outil important, parallèlement à l'amélioration de la gestion et de la sécurité, surtout là où de très grandes quantités de données ainsi que de l'information de très haute importance sont impliquées. Le DAM introduit plusieurs innovations dans TYPO3 n'ayant pas de lien direct avec sa fonction principale. L'introduction des *services*, ainsi que de nouvelles possibilités d'affichage dans le backend en constituent des exemples typiques.

En termes de fonctions, le DAM constitue la base pour le développement d'extensions proposant des fonctions de gestion de documents. En particulier, une passerelle entre le DAM et OpenOffice pour l'affichage et l'édition de documents pourrait mener à plusieurs développements augmentant l'utilisation de TYPO3 pour la gestion de contenu en entreprise. Le DAM a aussi

augmenté les chances d'introduire une manipulation uniforme des méta-données pour les fichiers (ressources (*Assets*)) et pour les éléments de contenu dans TYPO3. Cette fonction est particulièrement nécessaire pour répondre aux exigences du Semantic Web.⁹

4.15 Administration : l'avenir

On peut déjà prévoir que TYPO3 apportera plusieurs développements à l'administration telle qu'elle est décrite au début du chapitre (en plus de la gamme de fonctions déjà existantes). L'arrivée d'un système complet de gestion des versions dans TYPO3 est une étape importante que nous gardons en réserve. Le module de workflows sera amélioré sur cette base. À moyen terme, il deviendra important d'avoir un projet workflow d'envergure, dont le but sera de s'interfacer avec des moteurs de workflow externes. À long terme, le système d'administration des utilisateurs et d'autorisation des droits d'accès sera fortement influencé par les exigences d'un portail d'entreprise, dans la mesure où des travailleurs-clés peuvent s'engager dans ce développement, et où une contribution peut être apportée par des sponsors.

Référence 323622

C'est pourquoi les aspects relatifs à l'administration ont beaucoup d'influence sur le futur du noyau de TYPO3 ; ils représentent un tournant majeur qui transformera un WCMS en un système de gestion de contenu professionnel plus large et fondamentalement plus fonctionnel.

⁹<http://www.w3c.org/2001/sw/>

TYP03 pour les développeurs

5

Chapitre

TypoScript

5.1 Le rôle du développeur

5.1.1 Le processus de mise en œuvre

I think developers already know but are a little afraid to admit that writing software is a creative activity that requires a lot of interaction with the people who are going to use it.

— Richard Gabriel¹

Même si une plate-forme de communication basée sur TYPO3 demande beaucoup de temps et d'efforts, ce sont les étapes d'implémentation qui, au final, concrétisent l'objectif initial. Il est rare que ce processus ne connaisse qu'une solution. En effet, développer des interfaces et des fonctionnalités pour des utilisateurs laisse place à la créativité, mais également aux erreurs. La programmation n'est pas soumise uniquement aux lois du système ; grâce à un planning à long terme, elle doit aussi servir de point de départ à la maintenance, à l'extension et à la correction du code.

¹http://java.sun.com/features/2002/11/gabriel_qa.html

Deux points faibles étroitement liés au rôle du programmeur conduisent fréquemment à l'échec du développement d'une application :

- Si le but de l'application n'a pas été clairement expliqué aux personnes impliquées dans le projet, les descriptions de tâches seront axées sur les aspects techniques du logiciel. Aussi sophistiquées que soient les solutions, le succès d'une application dépend habituellement de questions simples : est-ce que les objectifs ont été clairement définis puis mis en pratique ? Le logiciel a-t-il été testé par des utilisateurs finaux pour vérifier sa simplicité et son ergonomie ? Finalement, toute application vit du contenu que les rédacteurs ajoutent rapidement et qu'ils ou elles gèrent. En d'autres mots : est-ce que le développeur a pensé à l'utilisateur ?
- Même la tâche de mise en œuvre est fréquemment sous-estimée. Lorsque les exigences sur l'implémentation augmentent, il faut envisager de faire appel à un sous-traitant qui ferait gagner du temps et donc de l'argent, sur base de son savoir-faire en la matière. En résumé : seule une personne connaissant bien les conditions et les possibilités du système conçoit des solutions pérennes qui, après qu'elles ont mûri, peuvent servir de base aux développements ultérieurs.

Ceci ne signifie pas que les outils pour créer de nouvelles applications TYPO3 ne peuvent être appris par n'importe qui — après tout, c'est précisément le but de ce livre.

5.1.2 Prérequis et vue d'ensemble

Si vous voulez élaborer des applications complexes avec TYPO3, une connaissance approfondie de TypoScript est un prérequis. TYPO3 fournit différents gabarits prêts à l'emploi avec lesquels même un débutant peut créer rapidement un site Web. Ces gabarits sont facilement installés et leur apparence (couleurs, espacement et logos) peut être modifiée à l'aide d'assistants. Mais si vous désirez faire des modifications ou implémenter certaines fonctionnalités en réponse aux exigences du projet, vous devez connaître TypoScript. Ce chapitre traite des principes de base du langage de configuration interne de TYPO3, TypoScript, et de son fonctionnement. Une certaine connaissance de l'HTML et du fonctionnement du *World Wide Web* est bien sûr nécessaire.

Une connaissance approfondie de PHP et de l'API de TYPO3 est, elle, requise uniquement si vous avez besoin de fonctionnalités qui dépassent le cadre de base et des nombreuses extensions disponibles gratuitement. Cet aspect est discuté en détail au chapitre 7.

Cette section du livre, "TYPO3 pour les développeurs" couvre les sujets suivants.

La section 5.2 est une introduction à TypoScript. Les enregistrements de gabarit et la possibilité de les mettre en cascade sont discutés, la syntaxe est expliquée, les principes de fonctionnement et d'emboîtement d'objets sont illustrés.

Les objets TypoScript, les fonctions, les types de données et le concept d'enveloppe (*wrap*) sont traités plus en profondeur à la section 5.3.

De nombreux outils de développement et d'aide sont introduits à la section 5.4.

Dans son installation de base, TYPO3 contient déjà une série de gabarits standards. La section 5.5 vous en donne une vue d'ensemble et décrit leur champ d'application.

Les sections 5.6 à 5.9 s'occupent de mettre en place une maquette de base. Plusieurs concepts de gabarits sont comparés et implémentés à titre illustratif dans un scénario. Une alternative à l'utilisation classique des gabarits ainsi que le changement de gabarit à l'aide de `type/typeNum` sont aussi passés en revue.

Le concept de navigation en TYPO3 implique différents types de menus basés sur du texte, des images, des couches, une image cliquable ou des menus de sélection. Leur configuration est expliquée à la section 5.10 à l'aide d'exemples spécifiques.

La section 5.11 s'intéresse plus particulièrement à certaines fonctions importantes telles que `stdWrap`, `optionSplit` et le GIFBUILDER qui jouent un rôle central dans TypoScript. En utilisant les propriétés et les conditions de `stdWrap`, vous pouvez insérer des structures de contrôle dans TypoScript. Le GIFBUILDER offre un large choix en termes de manipulation d'images.

Les sections 5.12 et 5.13 traitent de la conception d'un site avec des cadres et permettent d'entrevoir le futur avec des sujets tels que XHTML, l'accessibilité et TemplaVoilà.

5.2 TypoScript – Principes de base

5.2.1 Qu'est-ce que TypoScript ?

Un système de gestion de contenu doit pouvoir générer différents types de contenu et, grâce au principe de séparation entre le contenu et la forme, le mettre sous la forme voulue par chaque application. Une procédure largement utilisée est d'insérer des *balises* spéciales (`<tag-name>`) dans un *gabarit HTML statique* pour contrôler le résultat. De cette manière, le contenu spécifique, les fonctions, les listes ou les vues détaillées sont intégrés ou transformés vers le gabarit qui sera affiché.

Référence 387605

Avec l'introduction de *TypoScript* (TS), TYPO3 fait un pas supplémentaire puisque avec TypoScript vous pouvez créer le gabarit HTML dynamique vous-même, permettant un contrôle du format de sortie et de la disposition nettement plus avancé que cela n'est possible avec des gabarits HTML statiques.

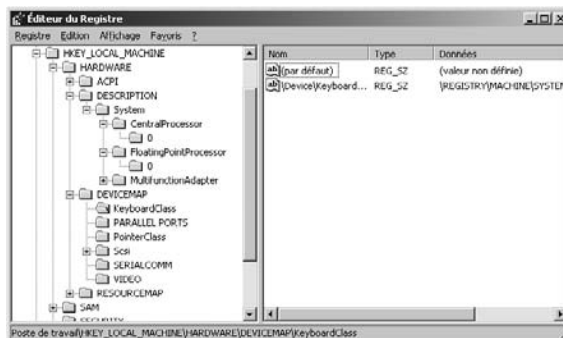
Avec l'information définie dans TypoScript, le contenu dynamique peut non seulement être inséré dans un gabarit, mais il est en plus possible d'influencer l'apparence (*frontend*) dans les moindres détails. La maquette graphique de base est générée complètement avec TypoScript ou est définie à partir d'un fichier HTML. Pour naviguer, vous utilisez un menu composé de textes, d'images, de combinaisons d'images, d'animation Flash ou d'une simple liste déroulante. Les menus sont générés dynamiquement et leur apparence est contrôlée individuellement. Vous pouvez créer à la volée les graphiques à partir de textes et d'images ou déterminer une fois pour toute leur disposition et leur contenu. De plus, TypoScript contrôle les options du backend afin de les personnaliser en fonction des utilisateurs ou des groupes d'utilisateurs. Comme nous l'avons mentionné au chapitre 4.1, vous spécifiez par exemple quelles sont les fonctions d'édition dont dispose le rédacteur dans le Rich Text Editor.

TypoScript joue le rôle de médiateur entre l'information et les fonctions qui sont développées en PHP dans le cœur de TYPO3, ou qui sont ajoutées par les extensions. De ce fait, TypoScript peut être vu comme une couche intermédiaire de transmission d'information aux fonctions du système.

Pour éviter tout malentendu, nous allons préciser ce que TypeScript n'est pas avant de définir ce qu'il est. La section 5.2.3 contient plus de détails techniques sur le sujet.

TypeScript n'est ni un langage de programmation ni un langage de scripting, et n'est donc pas comparable à Java, PHP ou JavaScript. Par exemple, il n'est pas possible d'utiliser de boucles (for, while,...). TypeScript sert en fait de « vecteur d'information ». Vous ne devez pas apprendre un langage nouveau, et surtout pas un nouveau langage spécifique. TypeScript lui-même n'est pas exécuté à n'importe quel moment.

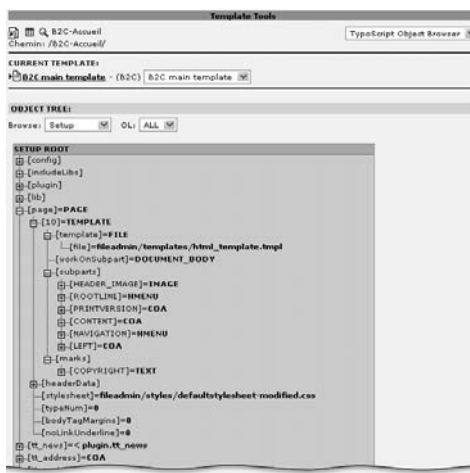
Figure 5.1:
Diagramme
hiérarchique des
objets et de leurs
valeurs structurées
par l'éditeur de
registre Windows



Si TypeScript n'est pas un langage de script, alors qu'est-ce ? Une description que vous devriez garder à l'esprit lorsque vous testez les limites de TypeScript est la suivante :

TypeScript possède une syntaxe permettant de définir de l'information de manière hiérarchique dans une structure arborescente simplement à l'aide de texte ASCII. De cette façon, des paramètres sont passés vers le système grâce à TypeScript, qui agit alors en tant qu'interface. Seuls les objets et les propriétés non définis dans le système n'influencent pas le fonctionnement du frontend et du backend. La documentation (TSref) correspondant à votre version de TYPO3 décrit l'ensemble des objets et des propriétés à votre disposition.

Figure 5.2:
TypeScript Object
Browser affiche la
structure des objets
et des valeurs



Les utilisateurs du système d'exploitation Windows se sont peut-être déjà familiarisés avec l'organisation hiérarchique des données dans le Registry, qui structure logiquement des valeurs à l'aide d'objets.

De manière similaire, les objets configurés par TypoScript sont organisés en une structure arborescente. L'outil de gabarit **TypoScript Object Browser** (le module **Web** → **Gabarit**) représente cette hiérarchie via une interface utilisateur graphique.

5.2.2 TSref

Nous voudrions à présent vous rappeler brièvement ce qu'est la référence TSref de TypoScript ; il s'agit d'une sorte de bible pour ceux qui travaillent quotidiennement avec TypoScript, puisque vous y trouvez des descriptions précises de tous les objets, propriétés et fonctions disponibles. Ayez toujours la référence à portée de main lorsque vous travaillez avec TypoScript ! TSref est disponible en ligne sur TYPO3.org et est disponible en format OpenOffice ou PDF. Si vous ne possédez pas encore TSref, le moment est venu de le télécharger !

Référence 342678

5.2.3 Digression : TypoScript et PHP

Afin d'acquérir une meilleure compréhension des aspects techniques, la section suivante traite de la relation entre TypoScript et PHP. Passez directement à la section 5.2.4 si vous le préférez.

Même si, en théorie, vous pouvez créer une sortie sous n'importe quelle forme en utilisant PHP, le *TypoScript Frontend Engine* (TSFE) propre à TYPO3 est utilisé par défaut lorsqu'un site Web est appelé via le fichier `index.php` (`tslib/index_ts.php`). Il analyse l'information dans l'enregistrement de gabarit de l'arborescence des pages du site Web².

PHP traite ici les objets et les valeurs des enregistrements de gabarit qui sont structurés par l'intermédiaire de TypoScript.

- L'information est placée par le système dans un tableau PHP multidimensionnel à l'aide de `t3lib_TSparger` (`t3lib/class.t3lib_tsparger.php`). Ce tableau est disponible dans certaines applications et fonctions de TYPO3.
- Si de l'information qui n'est pas utilisée par les fonctions des classes TYPO3 est placée dans le tableau PHP, elle se comporte comme une variable inutilisée déclarée en PHP : elle est ignorée et n'induit pas d'erreurs à la sortie.

Un exemple permet de clarifier cette notion : dans le code TypoScript abstrait suivant, l'information analysée par PHP est placée dans un tableau multidimensionnel.

```
monObjet.propriete1 = valeur_x
monObjet.propriete2 = valeur_y
monObjet.propriete2.propriete3 = valeur_z
```

En PHP, le tableau serait directement créé, comme suit :

```
$TS['monObjet.']['propriete1'] = 'valeur_x';
$TS['monObjet.']['propriete2'] = 'valeur_y';
$TS['monObjet.']['propriete2.']['propriete3'] = 'valeur_z';
```

²Le processus détaillé de restitution du frontend est décrit aux sections 5.7 et 7.5

ou alternativement :

```
$TS = array(
    'monObjet.' => array(
        'propriete1' => 'valeur_x',
        'propriete2' => 'valeur_y',
        'propriete2.' => array (
            'propriete3' => 'valeur_z'
        )
    )
)
```

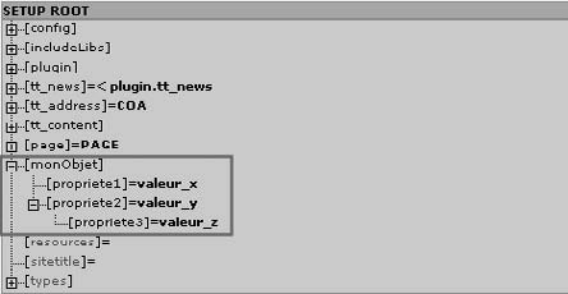
La fonction `debug()` propre à TYPO3 peut aussi afficher un tableau avec le résultat suivant :

Figure 5.3:
Affichage par la
fonction TYPO3
`debug()` de la syntaxe
Typo3

monObjet.	propriete1	valeur_x
	propriete2	valeur_y
	propriete2.	propriete3 valeur_z

TYP03 fournit un outil pour afficher et éditer TypeScript, appelé *TypoScript Object Browser*. Il affiche le code d'exemple de la manière suivante :

Figure 5.4:
Affichage du code TS
par le TypoScript
Object Browser



Exemple : HRULER

Référence 762761

Les effets combinés de TypoScript et de PHP sont illustrés par l'objet de contenu (cObject) **HRULER** qui trace une ligne horizontale. Considérons le gabarit TypoScript suivant :

```
page = PAGE
page.typeNum = 0
page.20 = HRULER
page.20 {
    lineThickness = 10
    lineColor = #e6e6e6
    spaceLeft = 100
    spaceRight = 100
}
```

À la troisième ligne, un objet TypeScript du type **HRULER** est défini ; des paramètres de configuration sont ajoutés dans les lignes suivantes. Les valeurs TypeScript des propriétés **lineThickness**, **lineColor**, **spaceLeft** et **spaceRight** sont placées par PHP dans un tableau. La fonction PHP **HRULER** (`tslib/class.tslib_content.php`) est alors disponible pour le traitement. Voici la fonction PHP³ :

```
function HRULER ($conf) {
$lineThickness = tslib_div::intInRange($this->stdWrap($conf['lineThickness'], $conf['lineThickness.']), 1, 50);
$lineColor = $conf['lineColor'] ? $conf['lineColor'] : 'black';
$spaceBefore = intval($conf['spaceLeft']);
$spaceAfter = intval($conf['spaceRight']);
$content = '';

$content.=' <table border="0" cellspacing="0" cellpadding="0" width="99% "><tr>';
if ($spaceBefore) {
    $content.='<td width="1"><img src= ».$GLOBALS['TSFE']->absRefPrefix.'clear.gif" width= ».$spaceBefore.'" height="1" alt="" /></td>';
}
$content.= '<td bgcolor= ».$lineColor.'"><img src= ».$GLOBALS['TSFE']->absRefPrefix.'clear.gif" width="1" height= ».$lineThickness.'" alt="" /></td>';
if ($spaceAfter) {
    $content.='<td width="1"><img src= ».$GLOBALS['TSFE']->absRefPrefix.'clear.gif" width= ».$spaceAfter.'" height="1" alt="" /></td>';
}
$content.= '</tr></table>';

$content = $this->stdWrap($content, $conf['stdWrap.']);
return $content;
}
```

Ceci produit une ligne horizontale d'une épaisseur de 10 pixels et de couleur **#e6e6e6**. De part et d'autre de la ligne, un espace d'une largeur de 100 pixels est inséré.

Cet exemple illustre clairement les possibilités de TypeScript, mais également ses limites. TypeScript offre aux développeurs une interface sûre pour configurer des fonctionnalités existantes, évitant les erreurs de manipulation de PHP et garantissant du code HTML correct à la sortie.

En même temps, la fonction précédente montre que le développeur n'a pas d'emprise sur le fait qu'une table (au sens HTML) est utilisée lors de l'affichage de **HRULER**, car cette caractéristique ne fait pas partie de la paramétrisation de cet objet.

Ce paradigme s'applique en principe à tout objet TypeScript : sa paramétrisation s'explique à partir des paramètres qui sont passés comme arguments à des fonctions PHP.

5.2.4 Gabarits TypeScript

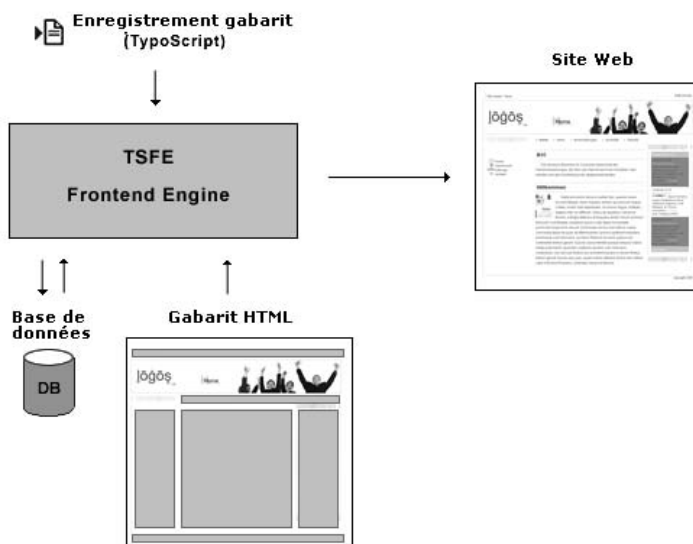
Les gabarits TypeScript déterminent la manière dont le TypeScript Frontend Engine restitue

Référence 917652

³Pour afficher une ligne horizontale, TYPO3 utilise une table, parce que les balises **<HR>** ne sont pas suffisamment contrôlées via CSS dans les navigateurs plus anciens.

le contenu, c'est-à-dire qu'ils déterminent quel contenu est lu dans la base de données, est-ce qu'un gabarit HTML est utilisé, où le contenu est inséré, etc. En outre, la transformation avant sa mise en forme dans le frontend est contrôlée par un gabarit TypoScript. En général, cela signifie que le gabarit décide quelles familles de fontes, quelles tailles de fontes, quelles couleurs et quels espacements sont utilisés dans le site Web.

Figure 5.5:
Une vue d'ensemble
du processus de
restitution du
frontend



Les gabarits TypoScript contrôlent les aspects suivants dans le processus de restitution du contenu :

- Le cache
- Le fichier journal (log)/les statistiques
- Les détails de l'en-tête HTML
- Les types de page (par exemple pour l'impression)
- L'agencement graphique de base (layout)
- Les éléments de contenu (apparence et fonction)
- La création de liens
- L'intégration d'extensions et de scripts PHP

TYP03 fournit des gabarits prédéfinis en fonction du domaine abordé, de sorte que les développeurs ne doivent pas spécifier pour chaque site Web toutes les définitions requises pour l'édition du contenu – comme la création de liens, par exemple.

Avant d'expliquer ces concepts, voici quelques exercices utilisant TypoScript, pour vous aider à vous familiariser avec ses bases.

5.2.5 Hello World ! – Le premier gabarit TypoScript

Pour que TYPO3 puisse restituer le contenu, un gabarit TypoScript est nécessaire. S'il manque, le message d'erreur « No template found ! » s'affiche lorsque la page est appelée dans le frontend.



Figure 5.6:
Gabarit TypoScript
manquant

Les gabarits TypoScript sont enregistrés dans l'arborescence des pages. Il y a deux types de gabarits : les *gabarits racines* (*root templates*) et les *gabarits d'extension* (*extension template*). Alors que les gabarits racines ont la case **Rootlevel** activée et se situent à la racine d'un site Web, les gabarits d'extension se trouvent partout dans l'arborescence des pages et étendent ou modifient les gabarits racines. Les enregistrements de gabarit peuvent donc être mis en cascade via l'arborescence des pages, ce qui signifie que TypoScript peut fusionner ou remplacer des gabarits.

Il y a deux manières de créer un nouveau gabarit : en cliquant sur l'icône d'une page dans l'arborescence des pages, vous créez un enregistrement du type **Gabarit** via l'option **Nouveau** du menu contextuel. L'enregistrement est immédiatement assigné à la page sélectionnée et il peut être édité comme les autres enregistrements.

La seconde manière est d'utiliser le module **Gabarit** → **Info/Modify**. Choisissez la page que vous voulez comme page racine d'un site Web (par exemple « Accueil ») dans l'arborescence des pages. Si aucun gabarit n'est encore assigné à la page, on vous l'indiquera et vous pourrez en créer un en l'insérant dans le champ du formulaire **Create new website** avec le bouton **Create template for a new site**. En option, vous pouvez intégrer un gabarit **Standard template** pour l'utilisation du site Web via le menu de sélection. Nous en reparlerons plus tard.

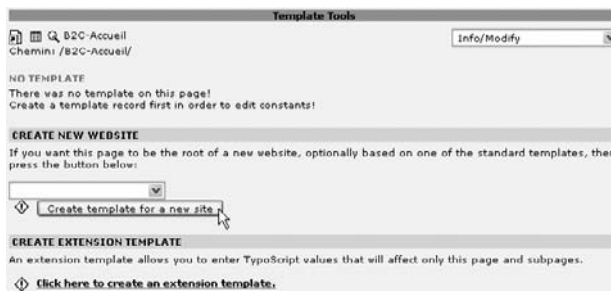


Figure 5.7:
Un nouveau gabarit
TypoScript avec le
module Gabarit →
Info/Modify

En supposant que vous ayez créé votre premier gabarit racine avec **Create template for a new site**, TYPO3 aura créé un enregistrement de gabarit avec comme titre « NEW SITE ». Six lignes de code TypoScript sont déjà contenues dans le champ **Setup**. Cette configuration TypoScript définit la sortie dans le frontend de la page contenant le texte « HELLO WORLD ! ».

Figure 5.8:
Gabarit affiché dans
le module Gabarit →
Info/Modify

TEMPLATE INFORMATION:	
NEW SITE	
Title:	NEW SITE
Site title:	
Description:	
Resources:	
Constants:	(edit to view, 0 lines)
Setup:	(edit to view, 6 lines)

[Click here to edit whole template record](#)

Il est possible d'éditer l'enregistrement complet via le lien en bas de page ou d'ouvrir une sélection des champs les plus importants avec les icônes correspondantes. Le champ **Setup** contient les détails suivants :

```
# Default PAGE object:
page = PAGE
page.typeNum = 0
page.10 = TEXT
page.10.value = HELLO WORLD!
```

Si vous éditez des champs individuellement, le cache de l'enregistrement est supprimé lorsque vous sauvegardez, et les dernières modifications sont prises en compte lors de la restitution de la page. Si vous éditez tout l'enregistrement, la mise à jour se fait par le bouton **Clear all cache** au bas de l'aire du module **Gabarit**, ou dans le frontend via l'Admin Panel. Si les détails ne s'affichent pas immédiatement lorsque vous développez un site, cela ne veut pas nécessairement dire qu'il y a des erreurs de données. Dans ces cas-là, effacez d'abord le contenu du cache.

Si l'enregistrement de gabarit est ouvert via le bouton **Click here to edit whole template record**, vous aurez une vue d'ensemble de toute l'information contenue dans chaque champ du formulaire. Notez que **Rootlevel** a été introduit dans le gabarit suite à l'action **Create template for a new site** et sert de point de départ du site Web. De plus, les détails de **Clear Constants** et **Clear Setup** qui ont été placés jusqu'à présent dans l'arborescence des pages et dans les gabarits TypoScript sont ignorés. Ces trois paramètres permettent de définir de nouveaux sites Web à partir de n'importe quel endroit de l'arborescence des pages.

Figure 5.9:
Vue partielle de
l'enregistrement de
gabarit

Chemin: /B2C-Accueil/

Gabarit (137) - NEW SITE

Template title:
NEW SITE
Destivated: Start: Stop:

Website title:

Constants:

Setup:
Default PAGE object:
page = PAGE
page.10 = TEXT
page.10.value = HELLO WORLD!

clear:
☒ Constants ☒ Setup

Rootlevel:

Si la page est appelée dans le frontend, vous verrez le résultat suivant :



Figure 5.10:
Affichage dans le
frontend

Pour mettre en forme le texte, ajoutez une ligne contenant les balises `<strong>` `</strong>` à la configuration TypoScript, comme ci-dessous :

```
# Default PAGE object:
page = PAGE
page.typeNum = 0
page.10 = TEXT
page.10.value = HELLO WORLD!
page.10.wrap = <strong>|</strong>
```

La sortie du frontend montre désormais « HELLO WORLD ! » en gras :



Figure 5.11:
La sortie du frontend
avec la configuration
modifiée

Les possibilités de TypoScript ne se limitent bien sûr pas à ce petit exemple. Mais illustrons d'abord les fonctionnalités des gabarits.

5.2.6 Cascade de gabarits

Dans l'exemple suivant, nous créons un second gabarit, « mon contenu », qui vient s'ajouter au gabarit « NEW SITE ». Pour ce faire, utilisez par exemple l'option **Create an extension template** du module **Gabarit** → **Info/Modify**.

Dans un gabarit d'extension, les cases à cocher **Rootlevel**, **Clear Constants** et **Clear Setup** ne sont pas actives par défaut. Un tel gabarit peut être placé où vous le désirez dans l'arborescence des pages. Cela permet d'une part d'associer du TypoScript à certaines arborescences de pages spécifiques, puisque le gabarit s'active automatiquement sur la page et ses sous-pages. D'autre part, vous pouvez placer les gabarits dans un Dossier Système pour les garder comme bibliothèque et pour les intégrer à d'autres enregistrements de gabarits. De cette manière, les gabarits sont mieux structurés, les fonctionnalités précises restent séparées et disponibles en tant que composants ; de plus, le code est réutilisable.

Dans les deux cas, vous emboîtez des enregistrements de gabarit. Cette procédure s'appelle aussi la *mise en cascade*.

L'intégration du gabarit « mon contenu » au gabarit « NEW SITE » se fait via le champ **Include basis template** de l'enregistrement de gabarit. Vous devez ouvrir l'enregistrement de gabarit

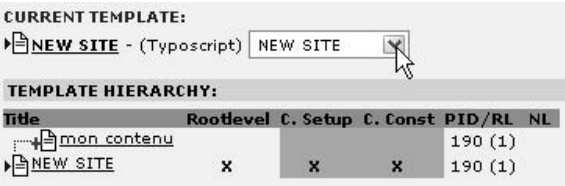
complet « NEW SITE » pour l'éditer. Pour sélectionner et intégrer le gabarit « mon contenu », utilisez le navigateur d'éléments.

Figure 5.12:
Intégration d'un
gabarit de base dans
« NEW SITE »



Le contrôle de la hiérarchie des gabarits se fait dans le **Template Analyzer**. Si vous sélectionnez le gabarit principal, ses dépendances seront montrées dans une structure arborescente. L'ordre, de haut en bas, correspond à l'ordre dans lequel les gabarits sont traités par le TypoScript Frontend Engine.

Figure 5.13:
Le Template Analyzer
du module Web →
Gabarit



Par exemple, l'objet `temp.monContenu` est défini dans le gabarit « mon contenu ».

```
temp.monContenu = TEXT
temp.monContenu.valeur = exemple de texte
```

Cet objet est donc disponible dans le gabarit « NEW SITE » et peut y être réutilisé, grâce à la cascade de gabarits.

```
page.20 < temp.monContenu
```

5.2.7 Enregistrements de gabarits

Référence 917198

Comme nous l'avons illustré dans la section précédente, un enregistrement de gabarit est absolument essentiel à la restitution du contenu dans le frontend. Un gabarit donné peut contenir l'information suivante :

Template title

Dans le champ **Template title**, le nom du gabarit est celui qui est affiché dans la vue d'ensemble du backend. Choisissez le nom que vous désirez. Toutefois, le nom devrait rendre compte des fonctions définies dans le gabarit puisque plusieurs gabarits différents sont souvent utilisés. Les cases à cocher **Deactivated**, **Start** et **Stop**, permettent de désactiver le gabarit de manière soit permanente, soit pour une durée déterminée, comme pour d'autres types d'enregistrements de TYPO3.

Website title

Spécifiez le titre du site Web dans le champ **Website title**. Dans la configuration par défaut, il s'insère dans l'en-tête HTML du frontend, avant le titre de la page, et dans la barre de titre de la fenêtre de votre navigateur :

```
<title>B2C: Produit</title>
<title>[Titre du site web]: [Titre de la page]</title>
```

Constants

Entrez dans le champ **Constants** les valeurs qui remplacent les constantes auxquelles le code inséré dans le champ **Setup** fait référence. Les constantes représentent des valeurs définies pour l'ensemble du site Web qui devraient être faciles à changer pour modifier en une fois des paramètres tels que la taille des fontes, les fontes, les couleurs de l'arrière-plan, ... Ne les confondez pas avec les variables telles qu'elles sont définies dans les langages de programmation. Les constantes sont remplacées par leurs valeurs dans l'ordre dans lequel elles sont entrées, et peuvent donc se remplacer l'une l'autre.

Setup

Ce champ contient le code de configuration TypeScript, qui définit l'apparence et le comportement de l'application. Les constantes (valeurs globales ou propriétés variables) apparaissant dans ce champ sont remplacées par les valeurs correspondantes définies dans le champ **Constants**.

Ressources

Ce champ contient des *ressources* c.-à-d. des images, des masques, des fontes True-type, des feuilles de style, de l'HTML et des documents texte. Pour faire une référence à ce champ avec TypeScript, utilisez le type de données **resource**. Lorsque vous copiez un gabarit, les ressources du champ **Resources** sont également copiées et numérotées séquentiellement. Par exemple, **logo.gif** devient **logo_01.gif**. Il est donc judicieux de faire référence aux ressources à l'aide du symbole (**logo*.gif**).

Clear et Rootlevel

Les cases à cocher **Clear Constants**, **Clear Setup** et **Rootlevel** forment un groupe fonctionnel qui intervient dans le contexte des gabarits en cascade.

Rootlevel

Rootlevel définit l'endroit dans la structure de la page qui est pris comme point de départ (racine) d'une nouvelle application (un site Web). Toutes les sous-pages (pages-enfants) dans l'arborescence des pages héritent des propriétés de chaque gabarit. Si de nouveaux gabarits sont associés à certaines pages, leur code TypeScript est fusionné au code des gabarits des pages définies à un niveau supérieur (pages-parents). De cette manière, vous pouvez surcharger à un niveau plus bas des valeurs qui ont été définies dans les gabarits-parents.

Le gabarit racine sert de point de départ à toutes les configurations d'une application en TypeScript jusqu'à ce qu'un gabarit définisse un nouveau Rootlevel. Un gabarit racine est indiqué avec une flèche bleue sur son icône.

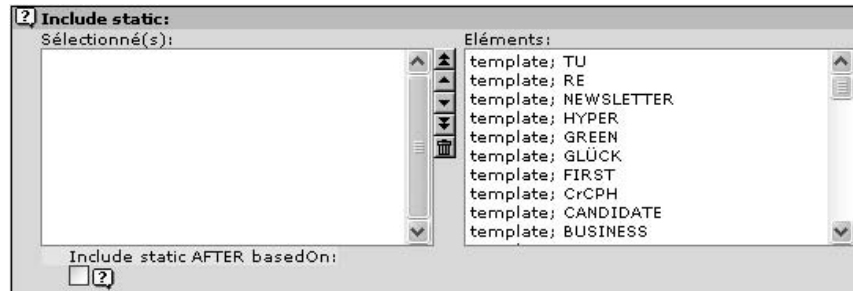
Clear Constants et Clear Setup

Vous pouvez, grâce à ces cases à cocher, arrêter la mise en cascade des **constants** ou du **setup** de l'arborescence des pages. Un gabarit défini de cette façon n'hérite d'aucune propriété **constants** ou **setup** des gabarits associés à des pages d'un niveau supérieur.

Include static

Une série de gabarits standards (*static templates*) est fournie par TYPO3 dans sa configuration de base. Ceux-ci comprennent non seulement toute la configuration TypoScript pour représenter des éléments de contenu, mais aussi, des maquettes graphiques de mise en page prêtes à l'emploi. Les différents types de gabarits standards sont introduits à la section 5.5.

Figure 5.14:
Choix de gabarits
standards

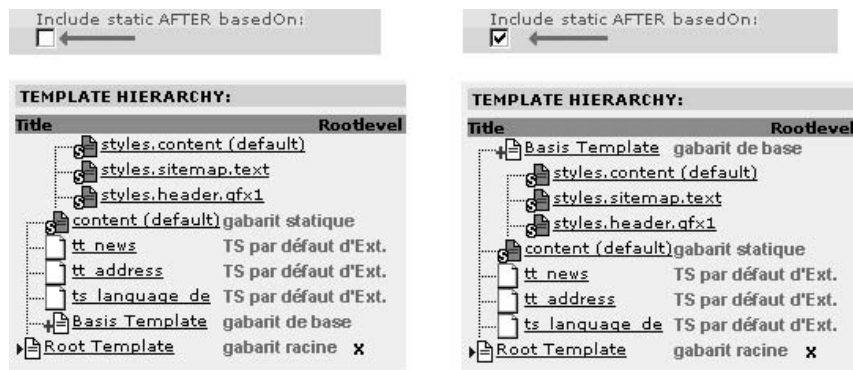


Si vous avez inclus plusieurs gabarits standards, vous déterminez l'ordre de traitement en cliquant sur l'icône à double flèche. Vous les supprimez à l'aide de l'icône « corbeille ».

La case à cocher **Include static AFTER basedOn** inverse l'ordre de traitement des gabarits de base (voir plus loin : **Include basis template**) et des gabarits standards par le TypoScript Engine. Dans la configuration par défaut, les gabarits de base sont traités après les gabarits standards. C'est le comportement classique puisque, de cette manière, vous écrasez des valeurs dans les gabarits standards par vos propres valeurs.

Dans l'exemple, un gabarit de base et le gabarit standard **content (default)** sont intégrés au gabarit racine. Les configurations TypoScript qui contiennent des extensions (**default TypoScript from Extensions**) se comportent comme des gabarits standards. Le Template Analyzer indique l'ordre dans lequel les gabarits sont traités, de haut en bas, et leur structure.

Figure 5.15:
Template Analyzer
montre l'effet de
Include static AFTER
basedOn



Include static (from extensions)

Les extensions contiennent aussi parfois des gabarits standards. Ces derniers sont intégrés dans la zone **Include static (from extensions)** du formulaire. Les configurations de TypeScript contenant l'extension de `ext_typescript_*.txt` (**default TypeScript from Extensions**) sont automatiquement chargées pendant l'installation. **General Office Displayer** et **CSS Styled Content** sont des exemples d'extensions qui contiennent des gabarits standards.



Figure 5.16:
Intégration des
gabarits standards
d'une extension en un
clic de souris

Include basis template

Les gabarits de base représentent les bibliothèques personnelles grâce auxquelles le développeur organise son code TypeScript de manière modulaire. Les gabarits de base permettent de gérer TypeScript de manière plus claire et permettent aussi d'encapsuler le code selon les fonctionnalités utilisées dans de futurs projets. Ils représentent leur propre enregistrement de gabarits pour lesquels les options **Clear Constants**, **Clear Setup** et **Rootlevel** n'ont pas été sélectionnées. Des gabarits de base peuvent inclure à leur tour leurs propres gabarits de base, formant ainsi une structure en cascade. Leurs icônes contiennent le signe « plus » vert.

Dans l'exemple ci-dessous, un gabarit nommé *Basis Template* est intégré au gabarit *Root Template*.



Figure 5.17:
Intégration d'un
gabarit de base

Static template files from T3 Extensions

Lorsque le Typoscript contient du code associé à des extensions (**Default TypeScript from extensions**), l'ordre d'intégration est aussi important. Combiné avec la case à cocher **Include static AFTER basedOn**, le champ de sélection **Static template files from T3 extensions** offre de nombreuses possibilités de mise en cascade des gabarits.

Default (Include before if Root-flag is set)

Cette option a pour effet d'insérer les gabarits standards d'extension avant le gabarit racine. Cela est utile pour remplacer les valeurs des gabarits d'extension par le gabarit racine. La différence entre cette configuration et la suivante est que les gabarits d'extension sont uniquement intégrés avant si **Rootlevel** est activé.

Always include before this template record

Lorsque vous optez pour ce choix, les gabarits standards de l'extension sont traités

juste avant les gabarits correspondants. Notez que c'est la seule manière de modifier le gabarit des extensions.

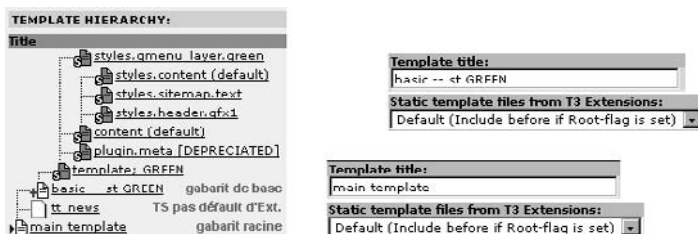
Never include before this template record

Cette option empêche les gabarits standards de l'extension d'être lus directement avant le gabarit considéré. Ceci signifie que bien que le gabarit de l'extension soit actif, il ne peut être modifié par une autre configuration TypoScript.

Dans l'exemple qui suit, un gabarit de base (basic – st Green) est intégré au gabarit racine (main template). Un gabarit standard (template; GREEN) est ensuite lui-même intégré au gabarit de base. L'extension News (tt_news), qui comprend son propre code standard TypoScript, est installée. L'ordre judicieux dans lequel les gabarits devraient être lus est le suivant : d'abord les gabarits standards, ensuite le gabarit pour les extensions, puisque ce dernier peut remplacer des valeurs des gabarits standards, suivi du gabarit de base pour remplacer certaines valeurs de ses prédécesseurs, et enfin le gabarit racine.

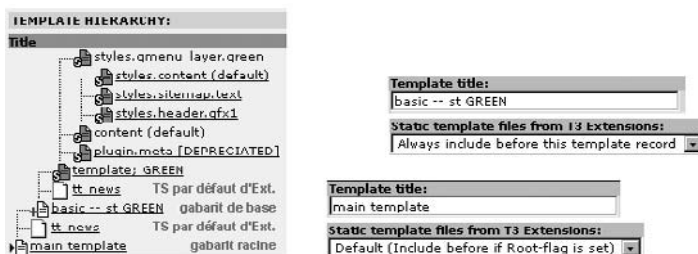
Dans le premier cas, le réglage de base **Default (Include before if Root-flag is set)** est laissé tel quel pour le gabarit racine et pour le gabarit de base qui y est intégré. Le code TypoScript de l'extension tt_news est lu après basic – st Green et avant main template.

Figure 5.18:
Static template files
from T3 Extensions –
Exemple 1



Si le réglage **Always include before this template record** est activé et que le réglage par défaut est conservé pour le gabarit racine, le code TypoScript de l'extension tt_news est lu deux fois – ce qui n'a pas beaucoup de sens.

Figure 5.19:
Static template files
from T3 Extensions –
Exemple 2



Dans le dernier exemple présenté ci-dessus, le réglage **Never include before this template record** pour le gabarit racine empêche le code TypoScript des extensions d'être lu juste avant main template. D'autre part, le gabarit de base force la lecture de ces données du TypoScript des extensions juste avant basic – st Green.

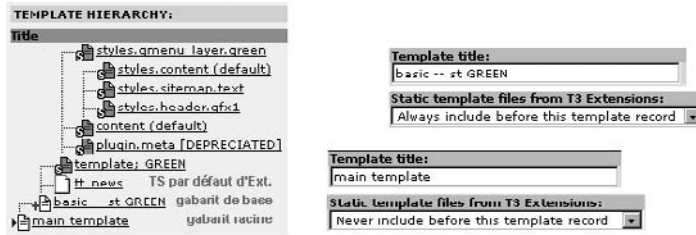


Figure 5.20:
Static template files
from T3 Extensions –
Exemple 3

Template on next level

Ce champ de sélection fournit un moyen simple d'intégrer un gabarit valide pour toutes les pages du niveau suivant. Ainsi, vous ne devez pas assigner un enregistrement de gabarit à chaque page du niveau suivant. Des changements de mise en forme pour des parties de l'arborescence sont donc rapidement insérés.

Description

Décrire un gabarit et ses fonctionnalités vous aide (vous ou une tierce personne) à vous retrouver par la suite dans le projet.

Backend Editor Configuration

Ce champ permet de faire passer des styles prédéfinis à l'éditeur CSS. Cette option n'est que rarement utilisée et sa maintenance n'est plus assurée.

Référence 626488

5.2.8 Constants et Setup

Les deux champs les plus importants d'un enregistrement de gabarit sont les champs **Setup** et **Constants**. **Setup** contient la configuration TypeScript, c'est-à-dire le code TypeScript contrôlant l'apparence et le paramétrage du site Web. Ce code est traité lors du processus de restitution du contenu.

De son côté, le champ **Constants** fait passer dans le champ **Setup** des constantes, c.-à-d. des valeurs facilement manipulables ou des valeurs globales. De cette façon, vous avez une vue d'ensemble lorsque vous modifiez des valeurs pour des pages en particulier. Par exemple, vous pouvez changer la valeur d'une couleur utilisée dans différents gabarits à partir d'un point central, au lieu de devoir la rechercher dans tous les gabarits utilisés.

La notation correspond à la syntaxe TypeScript normale : dans le champ **Setup**, les constantes sont entourées de `{ $ et }`.

Dans l'exemple suivant, « Hello TYPOS !! » s'affiche dans le frontend.

Constants :

```
monTexte.contenu = Hello TYPOS!!
```

Setup :

```
page = PAGE
page {
  typeNum = 0
  10 = TEXT
```

```
10.value = {$monTexte.contenu}
}
```

S'il n'y a pas de constante définie dans le champ **Constants** correspondant à celle appelée dans **Setup**, alors `$monTexte.contenu` s'affiche dans le frontend.

La hiérarchie des objets des données reprises dans les champs **Setup** et **Constants** est représentée clairement dans le TypeScript Object Browser (module **Web** → **Gabarit**). Pour ce faire, sélectionnez dans l'arborescence des pages celle dont vous voulez afficher le contenu du champ **Setup** et sélectionnez le mode avec **Browse : Constants** ou **Setup**.

Figure 5.21:
Affichage des
constantes dans
l'Object Browser



Faire passer du contenu par des constantes n'est bien entendu pas très sensé pour les pages dynamiques. Il est plus intéressant d'utiliser les constantes pour des valeurs telles que les formats et les couleurs de texte, les propriétés des images, etc., c'est-à-dire pour des valeurs qui sont utilisées de manière répétée dans les applications et qui devraient être faciles à modifier.

5.2.9 Éléments et concepts

Le gabarit **NEW SITE** créé à l'aide du module **Gabarit** → **Info/Modify** contient les lignes de code suivantes qui affichent dans le frontend une page HTML avec le texte « HELLO WORLD ! » :

```
# Default PAGE object:
page = PAGE
page.typeNum = 0
page.10 = TEXT
page.10.value = HELLO WORLD!
```

Les détails sont traduits de manière générique par :

```
# Commentaire
monObjet = TYPE_OBJET
monObjet.PROPRIETE = valeur_1
monObjet.sousObjet = TYPE_OBJET
monObjet.sousObjet.PROPRIETE = valeur_2
```

Pour vous présenter les différentes notions, les termes et leur appellation sont repris dans un exemple avant d'être définis plus en détail.

```
# Default PAGE object:
```

La première ligne est un commentaire et est donc ignorée par TYPO3.

Dans la ligne suivante, `page` est défini par l'opérateur « = » comme un objet de type `PAGE`.

```
page = PAGE
```

Le nom `page` peut, en principe, être choisi librement. Toutefois, par convention, certains noms sont réservés à des objets de base précis. Alors que `page` est utilisé pour les pages à contenu normal, le nom `plugin` est réservé aux plugins (extensions).

```
page.typeNum = 0
```

Un objet a des propriétés, et des valeurs sont assignées à ces propriétés. `typeNum` est une propriété du type d'objet `PAGE` à laquelle est assignée ici la valeur 0. La séquence de caractères `page.typeNum` définit le *chemin* de la propriété.

Les lignes suivantes définissent `page.10` comme un objet de type `TEXT`. En TypeScript, des listes numériques (*tableaux*) sont souvent utilisées. Dans ce cas-ci, la liste numérique est une propriété du type d'objet `PAGE`. Pour les objets de type `PAGE`, cette liste peut à son tour contenir d'autres objets. Ici, `10` est défini comme un objet de type `TEXT`. `page.10` est le chemin menant à cet objet.

```
page.10 = TEXT
```

Gardez à l'esprit que de telles listes ne peuvent pas toujours faire référence à d'autres objets ; cela dépend du type de données auquel appartient la liste. Vous trouverez dans TSref, la référence TypeScript, et dans les extensions correspondantes de l'information pertinente à ce sujet.

```
page.10.value = HELLO WORLD!
```

La propriété `value` de l'objet `page.10` prend la valeur `HELLO WORLD!`.

Dans la suite, nous reprenons en détail les concepts :

Types d'objets

TYPO3 vous fournit toute une série de types d'objets prédéfinis que vous pouvez utiliser dans vos gabarits TypeScript, par exemple les types `PAGE`, `TEXT` et `IMAGE`. La plupart des types d'objets servent à la restitution du contenu dans le frontend ; ils sont appelés *cObjects* ou *objets de contenu*. Les autres types d'objets sont utilisés à des fins de configuration générale.

Objets

L'objet `page` a été défini comme un objet de type `PAGE`. Puisque `page` est au sommet de la hiérarchie, on l'appelle aussi un *objet racine* (*oplevel object*). Ces objets peuvent porter presque n'importe quel nom. Les noms réservés tels que `lib`, `config`, `constants`, `styles` ou `temp` sont documentés dans TSref (voir la référence ci-contre).

L'objet `page` pourrait aussi être appelé `monday`, mais utiliser un nom descriptif aide à clarifier le code. Si vous voulez définir une page contenant des cadres (frame), le nom `frameset` serait par exemple un bon choix.

Référence 110843

Propriétés

Un type d'objet, et par conséquent un objet de ce type, a des propriétés définies précisément et décrites dans le document TSref. Seules ces propriétés sont prises en compte par les objets. Dès lors, la ligne suivante n'a de toute façon aucun effet, puisque l'objet `PAGE` n'a pas de propriété appelée `value`.

Référence 839954

```
page.value = Hello World!
```

Aucune erreur n'est toutefois créée puisque TypeScript n'est jamais exécuté.

Les propriétés d'un objet ont elles-mêmes un type de données spécifique. Par exemple, `page.typeNum` est du type `int`, ce qui signifie que les seules valeurs valides sont des nombres entiers. Les types et leurs paramètres valides se trouvent dans TSref.

Opérateurs

Les opérateurs sont introduits un peu plus tard. Par exemple, un de ceux-ci est l'opérateur d'assignation « = ».

Chemin et chemin d'objet

On fait référence aux objets et aux propriétés par leur chemin. Le chemin est constitué des objets et propriétés déjà définis, séparés par un point.

`page.10` est le chemin d'objet menant à l'objet correspondant ;

`page.10.value` est le chemin menant à la propriété `value` de cet objet.

5.2.10 La syntaxe

Pour simplifier, la notation se résume à l'expression suivante :

```
[Chemin d'objet].[Propriété] [Opérateur] [Valeur]
```

Les règles syntaxiques suivantes s'appliquent à TypeScript :

- Les objets et les propriétés sont séparés par un point qui traduit en même temps la dépendance hiérarchique.
- TypeScript est sensible à la casse.
- Les constantes ont la forme `{ $nom }` et sont remplacées par une valeur avant que le contenu du champ Setup de TypeScript soit traité.
- Pour les noms d'objets et leurs propriétés, seuls les caractères A-Z, a-z, 0-9 ainsi que les caractères « - » et « _ » sont utilisés.
- Le texte compris entre le début de la ligne et l'opérateur forme le chemin menant à l'objet ou à la propriété. Il ne peut pas contenir d'espace.
- L'ordre de traitement des propriétés d'objets est défini par l'objet lui-même ; il n'est donc pas défini par l'ordre des lignes du setup TypeScript.
- Les listes numériques (`page.10`, `page.20`) sont traitées par ordre croissant.

Constantes

Les règles syntaxiques du setup s'appliquent aussi aux constantes, avec les restrictions suivantes :

- Les constantes ne sont pas des objets et n'ont donc pas de propriétés ; dès lors, aucune fonction du type `stdWrap` ou `if` n'est disponible. Elles sont néanmoins organisées hiérarchiquement.
- L'opérateur de référencement `=<` n'est pas disponible.
- Pour les valeurs s'étendant sur plusieurs lignes, vous ne pouvez utiliser l'opérateur `()`, comme dans le setup.

Valeurs

Pour les valeurs, les règles suivantes sont d'application :

- Elles ne sont pas entourées de guillemets.
- Elles débutent après l'opérateur et se terminent à la fin de la ligne.
- Si elles s'étendent sur plusieurs lignes, elles peuvent être groupées à l'aide de l'opérateur `()`.
- Les espaces précédant et suivant les valeurs sont supprimés lorsque celles-ci sont assignées.

Commentaires et blocs de commentaires

Vous pouvez ajouter des commentaires à un document TypeScript. Si vous n'êtes pas trop familiarisé avec les techniques de programmation, nous vous recommandons d'inclure à vos projets suffisamment de commentaires pour pouvoir vous retrouver facilement dans votre code, même après un long intervalle de temps.

Les commentaires d'une ligne sont marqués d'un `/` ou d'un `#` et précèdent les éléments de contenu.

```
# Default PAGE object:
/ un autre commentaire
```

Les blocs de commentaires de plusieurs lignes commencent par `/*` et se terminent par `*/`, placé au début de la dernière ligne de commentaire. Le signe terminant un commentaire est important puisque sans lui, le code suivant le commentaire sera traité lui aussi comme un commentaire.

```
/*
  Insérez ici un commentaire
  s'étendant sur plusieurs lignes.
*/
```

Opérateurs

{ }

Les opérateurs { } vous aident à écrire du TypeScript plus clairement et de manière plus compacte en vous permettant d'emboîter plusieurs propriétés.

L'exemple suivant produit exactement le même résultat que le précédent.

```
page = PAGE
page {
  typeNum = 0
  10 = TEXT
  10 {
    value = HELLO WORLD!
  }
}
```

Cet emboîtement commence par une accolade ({}). Tous les autres détails suivant cet opérateur sur la même ligne sont ignorés lors de l'*analyse syntaxique*. La première propriété subordonnée se trouve à la ligne suivante. L'espacement en début de ligne améliore la lisibilité. L'emboîtement de propriétés se termine par l'opérateur } qui doit être le premier caractère de la ligne, espaces vides non compris.

()

Les parenthèses () renferment des valeurs qui peuvent prendre plusieurs lignes de code. Ceci peut s'avérer utile si vous reprenez du texte qui est déjà structuré ou si vous voulez indiquer des valeurs plus clairement, par souci de lisibilité.

L'exemple a été légèrement modifié ci-dessous afin d'illustrer cet usage. L'objet 10 y est défini comme un objet de contenu HTML. Du code HTML, créé par exemple dans un éditeur HTML, est assigné à la propriété page.10.value.

```
page {
  typeNum = 0
  10 = HTML
  10.value (
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td width="25%" align="left" bgcolor="#003366">
          <span style="color:#FFFFFF">Petit exemple:</span></td>
        <td width="75%" align="center" bgcolor="#84A1E5">
          <span style="color:#FFFFFF">Insertion d'une valeur sur plusieurs
            lignes</span></td>
      </tr>
    </table>
  )
}
```

Est alors affiché dans le frontend :

Figure 5.22:
L'opérateur ()

Petit exemple:

Insertion d'une valeur sur plusieurs lignes

=

Le signe égal, =, sert à l'assignation. Pour définir un objet, on l'assigne à un type d'objet. Une valeur est assignée à une propriété. Toute la chaîne de caractères suivant l'opérateur (et sur la même ligne de code que lui) est traitée comme une valeur. Les espaces vides précédant et suivant l'opérateur sont supprimés.

```
page.10.value = HELLO WORLD!
# est identique à
page.10.value=HELLO WORLD!
```

< (Copier)

Grâce à l'opérateur <, vous pouvez copier un chemin dans un autre. Des propriétés et des objets entiers peuvent être copiés. Lorsque vous utilisez cet opérateur, les propriétés et les valeurs de l'objet sont copiées. Les objets, valeurs et propriétés qui existent déjà dans le chemin dans lequel vous les copiez sont remplacés.

Nous assignons ci-après deux nouveaux objets à l'exemple « HELLO WORLD ! ». L'objet `page.15` est défini comme un objet de contenu `HTML`, et un « saut à la ligne » est assigné à la valeur de sortie. L'objet `page.10` est copié dans `page.20`. À l'intérieur des accolades, vous accédez aussi au chemin d'objet en utilisant la syntaxe `20 < .10`.

```
page = PAGE
page {
  typeNum = 0
  10 = TEXT
  10.value = HELLO World
  15 = HTML
  15.value = <br />
  20 < page.10
# ou alternativement :
# 20 < .10
}
```

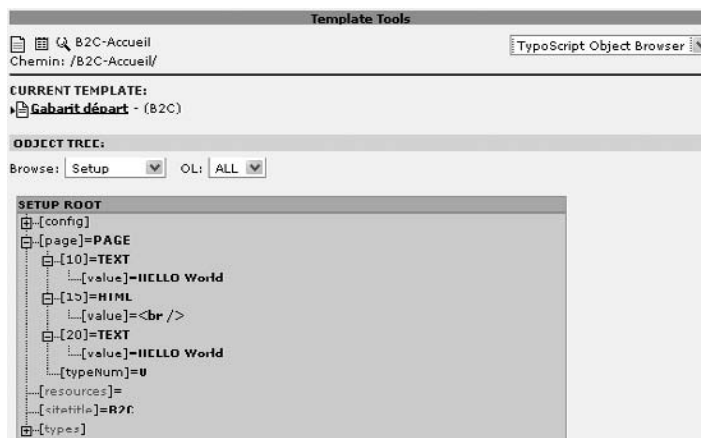
Puisque le même objet `TEXT` se retrouve dans `page.10` et `page.20`, le texte est affiché deux fois :

```
HELLO World
HELLO World
```

Figure 5.23:
Copie d'un chemin
d'objet par
l'opérateur <

Les objets et leur hiérarchie sont illustrés et contrôlés graphiquement dans le `TypoScript Object Browser` (**Web** → **Gabarit** → **TypoScript Object Browser** → [nom du gabarit]).

Figure 5.24:
TypeScript Object
Browser : les
propriétés de l'objet
10 sont copiées dans
l'objet 20



=< (Référencement)

Conjointement avec l'opérateur =, < ne copie pas le chemin d'objet mais y fait référence. Le référencement est possible uniquement avec les objets, pas avec les propriétés. Notez que ce cas-ci mis à part, il est impossible de combiner deux opérateurs en TypeScript.

L'exemple suivant illustre les différences de fonctionnement de la copie et du référencement :

```
# L'objet objetXY
objetXY = TEXT
objetXY {
    value = Hello TYPOS??
    textStyle.color.default = red
}

page = PAGE
page {
    typeNum = 0
    # L'objet 10 fait référence à l'objet objetXY.
    10 = < objetXY
    10.value = Hello TYPOS!! (référence)
    15 = HTML
    15.value = <br />
    # L'objet objetXY est copié.
    20 < objetXY
    20.value = Hello TYPOS!! (copie)
}

# Une propriété de l'objet objetXY est changée plus tard
objetXY.textStyle.color.default = blue
```

Dans le frontend, le texte de la copie est indiqué en rouge tandis que celui de la référence est indiqué en bleu (mais puisque l'illustration est en noir et blanc ci-dessous, vous devez nous croire sur parole...) :

Hello TYPOS!! (référence)
Hello TYPOS!! (copie)

Figure 5.25:
Comparaison entre
référencement et
copie

Avec l'opérateur `=<`, l'objet `page.10` pointe vers l'objet `objetXY` précédemment défini. Puisque l'objet `page.10` est une référence, le contenu de l'objet `objetXY` est repris à chaque fois que le TypeScript rencontre l'objet `page.10` dans son traitement. De cette manière, l'objet peut être utilisé plusieurs fois et à différents endroits de l'arborescence des objets, tout en étant toujours à jour puisque les changements dans l'objet référencé sont pris en compte dans toutes les références.

Dans le TypeScript Object Browser, vous pouvez vérifier comment le setup TypeScript a été assemblé. Le texte « Hello Typos !! (référence) » sera imprimé en bleu dans le frontend puisque la propriété `textStyle.color.default` de l'objet `objetXY` a été remplacée. Il est intéressant de remarquer que les propriétés de la référence peuvent être remplacées localement, ici en modifiant `page.10.value`. La copie dans le chemin d'objet `page.20` ne tient pas compte du changement dans la couleur du texte. En effet, les propriétés et valeurs de l'objet `objetXY` sont copiées lorsque le code TypeScript correspondant est lu, ce qui se produit avant que la couleur du texte ait été changée en bleu. C'est pourquoi le texte « Hello TYPOS !! (copie) » apparaît en rouge.

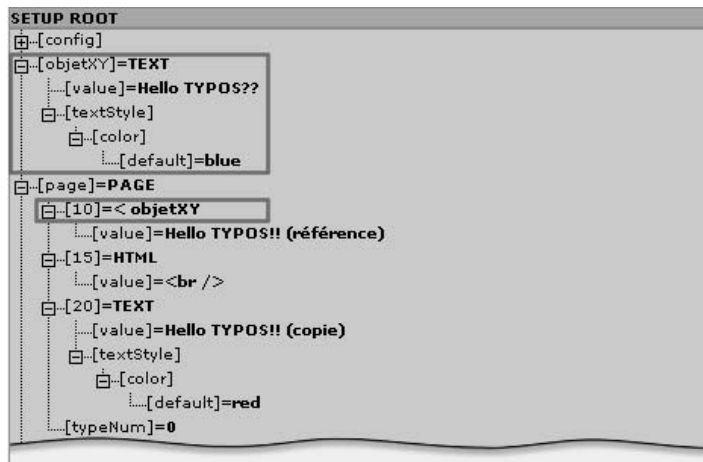


Figure 5.26:
TypeScript Object
Browser : l'objet 10
fait référence à l'objet
objetXY

>

L'opérateur `>` efface les objets et propriétés qui se trouvaient jusque-là dans le chemin donné.

Dans l'exemple suivant, l'objet `page.20` est supprimé.

```
page = PAGE
page {
  typeNum = 0
  10 = TEXT
```

```

10.value = HELLO World
15 = HTML
15.value = <br />
20 = TEXT
20.value = Hello TYPOS!!
20 >
}

```

Par conséquent, seule la première ligne de texte est affichée.

Figure 5.27:
« Hello TYPOS!! »
est supprimé

HELLO World

Conditions

Référence 270225

Les *conditions* dans TypoScript s'écrivent sur une seule ligne, sont entourées par des crochets ([]) et ne peuvent être placées à l'intérieur d'un emboîtement de propriétés (réalisé à l'aide d'accolades, { }). Pour que le code qui suit la condition soit lu, la condition doit être vérifiée. Ceci permet par exemple de restreindre les instructions à certains navigateurs ou groupes d'utilisateurs spécifiques.

Si l'analyseur syntaxique remarque que la condition est vérifiée, il continue d'interpréter le code TypoScript. Sinon, il ignorera le code jusqu'au marqueur de fin de condition.

Nous traitons les conditions en détail à la section 5.11. Nous nous intéressons ici uniquement à la syntaxe. Dans l'exemple suivant, la condition **[browser = netscape]** signifie que l'objet **page.10** est défini uniquement si le navigateur du visiteur du site est reconnu par TYPO3 comme étant un navigateur de type Netscape. La commande **[END]** (ou encore : **[GLOBAL]**) indique la fin de la condition.

```

page = PAGE
page.typeNum = 0

[browser = netscape]
page {
    10 = TEXT
    10.value = HELLO World
    10.wrap = |<br />
}
[END]

page {
    20 = TEXT
    20.value = Hello TYPOS!!
}

```

Lorsque plusieurs conditions se trouvent sur la même ligne, elles sont considérées comme étant séparées par l'opérateur logique OR, c'est-à-dire que tant qu'au moins une des conditions

est remplie, le code TypoScript suivant la condition sera lu. L'opérateur AND n'est pas encore supporté dans les conditions TypoScript.

Dans l'exemple qui suit, l'objet `page.20` affiche un texte différent si la requête provient d'un navigateur Netscape ou si le système d'exploitation est Linux.

```
[browser = netscape] [system = linux]
    page.20.value = vous êtes un utilisateur Netscape ou Linux
[END]
```

La condition `[ELSE]` est considérée comme étant vérifiée si la condition la précédant ne l'est pas. Cette condition se termine également par `[END]` ou `[GLOBAL]`.

```
page = PAGE
page.typeNum = 0
[browser = netscape]
page {
    10 = TEXT
    10.value = résultat pour le navigateur Netscape
}
[ELSE]
page {
    20 = TEXT
    20.value = résultat pour tous les autres navigateurs
}
[END]
page {
    30 = TEXT
    30.value = résultat pour tous les navigateurs
    30.wrap = <br />|
}
```

Par exemple, le navigateur Firefox est identifié comme un browser Netscape et produit donc le résultat suivant :

résultat pour le navigateur Netscape
résultat pour tous les navigateurs

Figure 5.28:
La condition `[browser = netscape]` est vérifiée

Notez que vous ne pouvez pas emboîter des conditions. Ainsi, l'exemple suivant ne fonctionne pas.

```
[browser = netscape]
    page.20.value = vous êtes un utilisateur Netscape
[system = linux]
    page.20.value = vous êtes un utilisateur Linux et Netscape
[END]
[END]
```

Inclusions

Vous pouvez inclure des bibliothèques TypoScript à partir de fichiers texte externes. L'instruction permettant ceci doit s'écrire sur une seule ligne mais peut, à l'inverse des conditions, se

trouver dans un emboîtement de propriétés. L'insertion et le traitement de la bibliothèque se font avant l'analyse syntaxique.

L'exemple ci-après vous montre comment intégrer un fichier externe à l'aide de l'instruction `<INCLUDE_TYPOSCRIPT>`.

```
page = PAGE
page.typeNum = 0
page {
  10 = TEXT
  10.value = résultat 1
  10.wrap = |<br />
  <INCLUDE_TYPOSCRIPT: source="FILE:fileadmin/typoscript/include_1.txt">
  30 = TEXT
  30.value = résultat 3
}
```

Le fichier externe `include_1.txt`, ayant été enregistré dans le système de fichiers sous `fileadmin/typoscript/`, contient le code suivant :

```
20 = TEXT
20.value = résultat 2 (intégré via INCLUDE)
20.wrap = |<br />
```

Le résultat suivant est affiché dans le frontend :

Figure 5.29:
Insertion de code
TypoScript par fichier

```
résultat 1
résultat 2 (intégré via INCLUDE)
résultat 3
```

5.2.11 Ordre de traitement

Pour utiliser TypeScript, vous devez comprendre l'ordre dans lequel le code TypeScript est traité.

1. Les instructions TypeScript dans le champ des constantes sont lues de haut en bas et les conditions sont prises en compte.
2. Les constantes sont remplacées dans le setup TypeScript dans l'ordre dans lequel elles ont été définies.
3. Les instructions TypeScript dans le champ setup sont lues de haut en bas.
4. Les conditions sont évaluées lorsqu'elles sont lues.
5. Les copies d'objets et de propriétés utilisant l'opérateur `<` se font lors de l'analyse syntaxique. Ceci signifie bien entendu que seule la configuration de l'objet disponible au moment de la copie sera copiée. Si des changements sont apportés à l'objet dans les lignes qui suivent, ils ne seront pas appliqués à la copie déjà créée.
6. Le code TypeScript est traité et les références sont intégrées.

Comme vous le voyez, la séquence des instructions durant l'analyse syntaxique de TypoScript est importante puisque la surcharge des propriétés dans les copies et les objets de sortie n'a pas d'effet sur les objets à partir desquels ils ont été copiés. D'autre part, l'ordre de traitement n'a pas d'importance.

- Les propriétés ne sont pas traitées dans l'ordre dans lequel elles sont créées dans le gabarit. C'est l'objet lui-même qui définit quand il évaluera une propriété.
- Les objets sont traités dans l'ordre de la liste numérique dans laquelle ils sont définis.

Ainsi, les exemples suivants donnent le même résultat ; il est évident que la première version est bien plus claire.

Exemple 1 :

```
page = PAGE
page.typeNum = 0
page.10 = TEXT
page.10.value = Je suis un
page.15 = HTML
page.15.value = <br />
page.20 = TEXT
page.20.value = texte exemple!
```

Exemple 2 :

```
page = PAGE
page.typeNum = 0
page.143 = HTML
page.143.value = <br />
page.377 = TEXT
page.377.value = texte exemple!
page.23 = TEXT
page.23.value = Je suis un
```

Je suis un
texte exemple!

*Figure 5.30:
Affichage des
exemples dans le
frontend*

5.2.12 L'emboîtement d'objets

Un autre principe essentiel du TypoScript est l'emboîtement d'objets. Jusqu'à présent, vous avez vu comment les objets créaient du contenu et l'affichaient dans l'ordre spécifié par une liste numérique. Le code de setup suivant en donne une illustration :

```
page = PAGE
page.typeNum = 0
page.10 = TEXT
page.10.value = je suis un
page.10.wrap = |<br />
```

```
page.20 = TEXT
page.20.value = petit
page.20.wrap = |<br />
page.30 = TEXT
page.30.value = texte exemple!
```

Ce code produit la sortie :

```
je suis un
petit
texte exemple!
```

L'hypothèse générale est que chaque objet à son tour se vide de son contenu de sorte que le résultat s'accumule graduellement dans le frontend. En fait, ce n'est pas tout à fait correct : les objets restituent leur contenu ou leur données dans l'objet qui les intègre. Dans l'exemple ci-dessus, les objets HTML transfèrent leurs éléments de contenu à l'objet PAGE situé à un niveau supérieur. Les éléments de contenu créés sont retournés, puis affichés, uniquement après que tous les objets de `page` ont été traités.

Cet emboîtement peut s'étendre sur plusieurs niveaux, étant donné que beaucoup de types d'objets peuvent eux-même intégrer des objets. Le code setup suivant donne le même résultat que l'exemple précédent :

```
page = PAGE
page.typeNum = 0
page.10 = COA
page.10 {
    10 = COA
    10 {
        10 = COA
        10 {
            10 = TEXT
            10.value = je suis un
            10.wrap = |<br />
        }
        20 = TEXT
        20.value = petit
        20.wrap = |<br />
    }
    20 = TEXT
    20.value = texte exemple!
}
```

Le même code n'utilisant pas d'accolades ressemble à ceci :

```
page = PAGE
page.typeNum = 0

page.10 = COA
page.10.10 = COA
page.10.10.10 = COA

page.10.10.10.10 = TEXT
```

```

page.10.10.10.10.value = je suis un
page.10.10.10.10.wrap = |<br />

page.10.10.20 = TEXT
page.10.10.20.value = petit
page.10.10.20.wrap = |<br />

page.10.20 = TEXT
page.10.20.value = texte exemple!

```

Pour comprendre ce setup TypoScript, il faut savoir que le type d'objets COA, de même que le type d'objets PAGE peut intégrer des objets dans une liste numérique. La procédure est la suivante : un objet COA est appelé par `page.10` ; à son tour, cet objet (via `page.10.10`) appelle un objet COA, et ainsi de suite. Par après, un premier objet de contenu (TEXT) est appelé par `page.10.10.10.10`, qui crée en réalité une sortie, à savoir « je suis un ». Tous les objets dans COA `page.10.10.10` ont maintenant été traités. L'objet COA a reçu le contenu généré à partir de l'objet TEXT incorporé et le passe à présent à son parent COA `page.10.10`. Ceci a pour effet de traiter sa liste d'objets en appelant l'objet TEXT suivant : `page.10.10.20`. Le contenu généré est annexé au contenu que le COA a reçu de l'enfant COA et, après que la liste d'objet a été traitée, l'objet PAGE reçoit les éléments de contenu, et ils sont affichés.

Cet emboîtement est illustré en ajoutant le code suivant :

```
page.10.10.stdWrap.case = upper
```

Ce qui a pour effet d'afficher la sortie :

```

JE SUIS UN
PETIT
texte exemple!

```

L'objet COA `page.10.10` reçoit, avec `.stdWrap.case=upper`, l'instruction de convertir son contenu en majuscules. L'objet lui-même ne crée pas de contenu mais, dans une hiérarchie emboîtée, incorpore le contenu des deux objets TEXT pour le convertir en majuscules.

Donc, des arborescences d'objets sont créées en TypoScript, où d'une part les niveaux sont traités tour à tour, et où, d'autre part, l'ordre de traitement dépend simultanément de la hiérarchie. De cette manière vous disposez d'un puissant outil pour configurer la restitution du contenu dans le frontend.

5.3 Objets, fonctions et types de données TS

5.3.1 Types de données

Les types de données utilisés par TYPO3 sont très variés et ne sont pas comparables aux types de données des langages de programmation. Bien qu'il y ait également des types de données comme `int`, `string` et `boolean`, TypoScript utilise aussi `degree` et `pixels`. Comme `int`, ce sont également des valeurs entières, à ceci près que `degree` définit un nombre de degrés et `pixels` un nombre de pixels. Comme vous le voyez, les types de données nous aident à décrire

Référence 253434

quelles sortes de données sont assignées à une propriété. Parfois, le type de données clarifie la fonctionnalité de la propriété. La liste complète des types de données se trouve à la référence ci-contre.

Un élément caractéristique des types de données réside dans le fait que certains d'entre eux sont en fait des fonctions. Jusqu'à présent nous vous avons indiqué de quelle manière TypeScript fixe les paramètres de configuration. Il utilise une procédure purement statique, sauf pour les conditions. Mais avec les fonctions disponibles, TypeScript devient un outil dynamique et peut traiter et modifier des données.

Voici un exemple très simple :

```
page.10 = HTML
page.10.value = kasper
page.10.value.case = upper
```

`value` est dans l'objet `HTML` et son type de données est `stdWrap`. Cette fonction connaît la propriété `case` et peut donc être configurée avec la valeur `upper` pour convertir la valeur en majuscules.

TypeScript fournit toute une série de fonctions similaires qui sont décrites plus en détail dans la suite. Ces fonctions combinées avec des types d'objets transforment TypeScript en un outil puissant.

Les types de données sont repris dans les tables des propriétés d'objets dans le document TSref. Dans une assignation, vous trouvez non seulement des types de données mais aussi des valeurs telles que `->select` ou `->filelink`.

Référence 924038 Dans ces cas-là, les fonctions sont appelées via les propriétés du même nom ou alors des objets prédéfinis du type précisé sont disponibles (par exemple TLO⁴ `config`). À la figure suivante, voyez comment les propriétés `if` et `stdWrap` pointent vers les fonctions correspondantes.

Figure 5.31:
Le type d'objet CASE
comprend les
fonctions `stdWrap` et
`if`

Property:	Data type:	Description:	Default:
setCurrent	string /stdWrap	Sets the "current"-value.	
key	string /stdWrap	This is the	
default	cObject		
Array...	cObject		
stdWrap	->stdWrap		
if	->if	If "if" returns false nothing is returned	
[tsref (cObject) CASE]			

Référence 034017 Si nous ajoutons `+calc` à la valeur d'un type de données dans TSref, vous pouvez alors appliquer des fonctions mathématiques sur cette valeur. Les opérateurs `++`/`*` utilisés dans ces calculs ne connaissent pas de priorités et sont traités dans l'ordre dans lequel ils apparaissent.

Référence 695668 Comme vous le voyez à la figure 5.31, la propriété est du type de données `string/stdWrap`. Pour des types de données joints comme ceux-ci, soit vous spécifiez une valeur (`string`), soit vous utilisez la fonction précisée (`stdWrap`). Vous pouvez également combiner les deux.

⁴Toplevel Object

5.3.2 Le concept d'enveloppe

Le concept d'*enveloppe* (wrap) est très important pour de nombreux objets TypoScript. Il est représenté par une chaîne de caractères divisée par un caractère (*tube*) « – ». Si l'objet a la propriété d'une enveloppe, il est entouré par la valeur de l'enveloppe.

Dans notre exemple, l'objet `page.10` de type `TEXT` avec la valeur `Hello World` est entouré par l'enveloppe `<strong>|</strong>` :

```
page.10 = TEXT
page.10.value = HELLO WORLD!
page.10.wrap = <strong>|</strong>
```

Le résultat est alors du texte en gras : `<strong>HELLO WORLD!</strong>`.

Les espaces, les tabulations et les sauts à la ligne avant et après les différentes parties de l'enveloppe sont supprimés lorsqu'ils sont lus.

5.3.3 Fonctions

Comme nous l'avons déjà décrit, certains types de données ont des capacités spéciales. Ils sont utilisés par de nombreux objets de contenu et leur fournissent des fonctionnalités universelles. Ils sont implémentés en PHP en utilisant des fonctions indépendantes, c'est pourquoi ils sont également repris comme fonctions dans TSref. Ils concordent toutefois parfaitement avec d'autres types de données en TypoScript pour s'insérer dans la syntaxe des objets et propriétés. Mais ils sont utilisés uniquement si les propriétés sont du type de données correspondant, puisqu'ils sont appelés explicitement au niveau du code PHP. Vous trouverez dans TSref le type de données de chaque propriété.

Référence 991290

Gardez à l'esprit que les fonctions ne sont pas disponibles dans le champ des constantes.

stdWrap

Cette fonction joue un rôle très important dans TYPO3. Selon l'usage que vous en faites, elle peut importer, contrôler ou manipuler des données. Une bonne partie des fonctions ci-après sont utilisées à partir de `stdWrap`. Donc, lorsque `stdWrap` est le type de données d'une propriété, des outils puissants sont à votre disposition pour le traitement des éléments de contenu.

imgResource

Cette fonction définit la source d'un fichier image, soumet l'image à un simple traitement et crée le code HTML requis pour la sortie. Elle est disponible aux propriétés appartenant au type de fichiers du même nom (`imgResource`).

imageLinkWrap

Cette fonction permet d'éditer une image à l'aide des éléments avancés de la bibliothèque graphique (GIFBUILDER). Vous pouvez en outre établir un simple lien (`typolink`) ou un lien avec le script `showpic.php`, pour ouvrir l'image dans une fenêtre JavaScript séparée.

numRows

Permet de déterminer le nombre d'enregistrements dans une table ou dans une requête de type `Select`.

select

Vous permet de formuler une requête SQL qui sera affichée par le cObject **CONTENT**. Les enregistrements cachés ou supprimés ne sont pas pris en compte dans ce processus, de même que ceux dont la visibilité est restreinte par Lancement/arrêt ou à des groupes d'utilisateurs frontend.

split

Cette fonction divise le contenu d'un texte sur base d'une lettre ou d'un caractère précis et traite les sections résultantes avec d'autres fonctions ou objets de contenu. Elle peut par exemple être utile si vous voulez qu'un menu utilise plusieurs couleurs.

if

La fonction **if** imite la structure de contrôle classique **IF**. Si les conditions subordonnées sont toutes remplies, les éléments de contenu de l'objet enfant restent telles quelles. Sinon, la fonction retourne **FALSE** et les éléments de contenu de l'objet sont supprimés.

typolink

typolink transforme le contenu en un lien. Cette fonction doit être utilisée autant que possible à la place des liens configurés "en dur", entre autres pour supporter la simulation des pages HTML statiques.

textStyle

La fonction **textStyle** quelque peu vieillissante définit l'apparence des zones de texte lorsque vous utilisez **FONT** ou d'autres éléments d'HTML. L'alternative moderne serait d'utiliser les définitions CSS.

encapsLines

Permet d'insérer des paragraphes, par exemple dans des balises HTML, pour la sortie. Des balises telles que **P** ou **DIV** sont ajoutées à l'analyse syntaxique et ne sont pas sauvées dans la base de données avec le contenu, ce qui fait que ce dernier reste « propre ».

tableStyle

Vous entourez le contenu d'une balise **TABLE** grâce à **tableStyle**. Différents détails sont définis ici. La fonction est utile, entre autres, en prévision de la table de mise en page, surtout pour configurer des gabarits TypoScript très soignés.

addParams

En utilisant cette fonction, vous passez des attributs (paramètres) supplémentaires aux éléments HTML de cObjects existants.

filelink

Un nom de fichier est converti en un lien de téléchargement grâce à **filelink**. Les détails du chemin, des attributs tels un texte ou un titre alternatif et des icônes sont ajoutés. Des aspects avancés de cette fonction comptent le nombre de téléchargements ou le camouflage du chemin d'accès au fichier sur le disque pour des raisons de sécurité.

parseFunc

Un post-traitement étendu des éléments de contenu est rendu possible par cette fonction. Vous pouvez par exemple remplacer les marqueurs insérés dans le texte, définir des règles d'utilisation ou de remplacement d'éléments HTML, ou encore démarrer un traitement récursif de blocs HTML emboîtés, tout en introduisant des fonctions supplémentaires. Vous pouvez ainsi envelopper les éléments de contenu ou des termes de recherche

de manière à ce qu'ils soient identifiés plus facilement, déterminer la manière avec laquelle les liens ou les listes (ordonnées ou non-ordonnées) sont générés, ou quelles balises peuvent apparaître dans les éléments de contenu.

makelinks

makelinks génère des liens HTML complets à partir de simples adresses Web ou email qui apparaissent dans le contenu.

tags

Utilisez cette fonction pour définir vos propres balises. Elles sont sauvegardées dans la base de données et sont remplacées à la sortie par différents éléments HTML qui existent réellement. Cette fonction est utilisée de pair avec **parseFunc**. Un exemple de balise définie par un utilisateur de TYPO3 est l'élément **<LINK>** qui est remplacé lors de l'analyse syntaxique par un lien HTML propre.

HTMLparser

La fonction **HTMLparser** sépare l'entrée HTML complète en de simples éléments HTML et leurs attributs. Ce faisant, les règles définies sont appliquées à la combinaison des éléments et des attributs. La fonction est surtout utile pour manipuler du contenu HTML importé ou extérieur, et lorsque vous travaillez avec le Rich Text Editor.

HTMLparser_tags

HTMLparser_tags est appelée par la fonction **HTMLparser** mentionnée ci-dessus. Elle définit des règles précises pour les attributs de chaque élément HTML.

5.3.4 Objets de contenu (cObjects)

TypoScript consiste principalement en des *Objets de Contenu* ou *cObjects* : ils restituent du contenu à partir de données en utilisant une configuration TypoScript. Bien que d'autres méthodes d'entrée et de sortie de données soient possibles, les objets de contenu sont généralement utilisés pour transformer en sortie frontend les données dont les types de contenu sont disponibles dans le backend (**Text**, **Image**, **Insert plugin**, ...) et sauves dans la base de données.

Référence 207606

Un tableau accessible de données courantes est assigné à chaque cObject lorsqu'il est traité. Il s'agit généralement des données de l'enregistrement actuel. Si un type de contenu **Text w/image** est restitué, les données entrées (par exemple l'en-tête sur les noms des fichiers des images insérées) sont disponibles pour la configuration TypoScript qui affiche cet élément de contenu parce que les champs de l'enregistrement sont accessibles (cf. section 5.11.3).

Dans les exemples précédemment exposés, nous avons uniquement utilisé les cObjects **TEXT**, **HTML** et **HRULER**. Mais il existe beaucoup d'autres cObjects en TYPO3.

Exemple : cObject FILE

Le cObject **FILE** intègre des éléments de contenu du système de fichiers ayant une taille allant jusqu'à 1024 kB. Notez que les images de type JPG, GIF, JPEG et PNG sont affichées avec une balise ****. Tous les autres formats sont lus et leur contenu est affiché sans traitement.

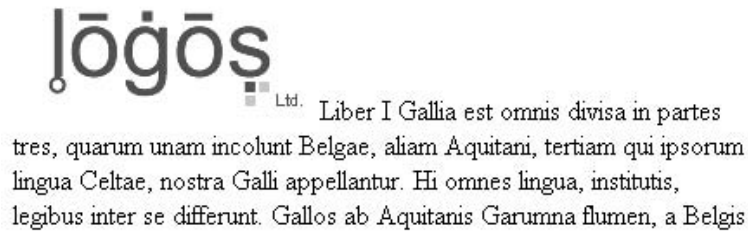
Référence 036048

Dans l'exemple qui suit, **page.10** est défini comme un cObject **FILE**. La propriété **file** (type de donnée : **resource**) pointe vers une image en format GIF. L'objet **page.20** est aussi défini

comme un cObject FILE mais pointe vers un fichier en format TXT. Le résultat est que ce fichier est lu et son contenu affiché.

```
page = PAGE
page {
  typeNum = 0
  10 = FILE
  10.file = fileadmin/images/layout/logo.gif
  20 = FILE
  20.file = fileadmin/documents/text1.txt
}
```

Figure 5.32:
Accéder au système
de fichiers avec le
cObject FILE (sortie
frontend)



Vue d'ensemble des cObjects

HTML

HTML est utilisé pour afficher du contenu HTML. La propriété `value` est du type de données `stdWrap` avec lequel les propriétés de la fonction du même nom sont disponibles.

Exemple :

```
10 = HTML
10.value = Ceci est un objet HTML
10.value.case = upper
```

TEXT

TEXT est très similaire au cObject HTML et est utilisé pour afficher du texte qui n'est pas mis en forme. Les propriétés de la fonction `stdWrap` sont appelées à partir de la base de l'objet, ce qui n'est toutefois pas la règle générale.

Exemple :

```
10 = TEXT
10.value = Ceci est un objet de texte
10.case = lower
```

COBJ_ARRAY (COA, COA_INT)

Le cObject COBJ_ARRAY (alternativement : COA) comprend plusieurs objets dans une liste numérique, dans un seul tableau. Si vous créez un objet du type COA_INT, il se

comportera exactement comme l'objet `USER_INT` : il est restitué sans passer par le cache !

Exemple :

```
temp.monObjet = COA
temp.monObjet {
  10 = HTML
  10.value = <table border=1 cellspacing=5 bgcolor=grey><tr><td>
  20 = TEXT
  20.value = Un objet, créé via COA.
  30 = HTML
  30.value = <td><tr></table>
}
```

FILE

Avec le cObject `FILE`, vous accédez au système de fichiers et intégrez du texte, des images ou des fichiers HTML par exemple.

Exemple :

```
10 = FILE
10.file = fileadmin/html/html-contenu1.htm
```

IMAGE

Les images peuvent être intégrées avec le cObject `IMAGE`. La propriété `file` du type de données `imgResource` permet d'accéder au système de fichiers ou aux ressources de l'enregistrement de gabarit.

Exemple :

```
10 = IMAGE
10.file = entete_commercant*.gif
20 = IMAGE
20.file = fileadmin/images/layout/entete_produit.gif
```

IMG_RESOURCE

Le cObject `IMG_RESOURCE` renvoie uniquement la référence à l'image, sans balise `<img>`.

Exemple :

```
10 = IMG_RESOURCE
10.file = entete_commercant*.gif
10.stdWrap.wrap = <table width="400" border=0 cellspacing=15
background="|" "><tr><td>Notre commerçant<td><tr></table>
```

CLEARGIF

Le cObjet `CLEARGIF` insère une image GIF transparente qui peut servir pour insérer des espaces blancs dans les gabarits.

Exemple :

```
10 = CLEARGIF
10.height = 15
```

CONTENT

Avec le cObject **CONTENT**, vous affichez les éléments de contenu de la base de données. La requête Select est restreinte aux tables commençant par **tt_**, **tx_**, **ttx_**, **fe_** ou **user_**. Pour afficher les enregistrements, vous avez besoin d'un TLO configuré de manière adéquate avec le nom de la table qui définit la sortie. La table **tt_content**, par exemple, est préconfigurée dans le gabarit standard **Content (Default)**.

Exemple : le résultat des éléments de contenu de la page courante et de la colonne « Normal ».

```
10 = CONTENT
10.table = tt_content
10.select {
    pidInList = this
    orderBy = sorting
    where = colPos=0
}
```

RECORDS

Avec le cObject **RECORDS**, vous pouvez afficher les enregistrements de la base de données, ce qui est comparable au type de contenu **Insert record**. Vous ne pouvez normalement pas sélectionner des enregistrements à partir de pages inaccessibles (cachées, restreintes dans le temps ou dont l'accès est protégé), en tout cas tant que l'option **dontCheck Pid** n'est pas configurée.

Exemple : les enregistrements d'adresses avec UID2 et 3 sont affichés.

```
10 = RECORDS
10 {
    source = tt_address_2,3
    tables = tt_address
    conf.tt_address = < tt_address.default
}
```

HMENU

Ce cObject vous permet de générer un menu hiérarchique. La propriété (1/ 2/ 3/...) fait référence au niveau du menu dans lequel l'objet sera restitué.

Exemple :

```
10 = HMENU
10.1 = TMENU
...
```

CTABLE

Le cObject **CTABLE** crée une table HTML dont les cellules de contenu sont entourées par quatre autres cellules. Le contenu peut être assigné à chacune de ces cellules.

Exemple :

```
10 = CTABLE
10 {
    tableParams = border=0 width=500
    c.10 = CONTENT
    ...
}
```

```

    rm.10 = HMENU
    ...
}

```

OTABLE

Cette table HTML est utilisée pour ajouter un décalage aux éléments de contenu. Concrètement, ils sont décalés à partir du coin supérieur gauche.

Exemple :

```

10 = OTABLE
10 {
    offset = 10,70
    tableParams = border=0 width=100
    10 = TEXT
    10.value = Contenu
}

```

COLUMNS

Avec ce cObject, vous créez une table pour laquelle vous spécifiez le nombre de colonnes et de lignes, les paramètres, la largeur, l'espacement des colonnes et l'épaisseur des bordures.

Exemple :

```

10 = COLUMNS
10 {
    1 = CONTENT
    ...
    2 = CONTENT
    ...
    gapWidth = 30
    gapLineThickness = 1
    if.isTrue.numRows < .1
    if.isTrue.ifEmpty.numRows < .2
    totalWidth = 500
}

```

HRULER

Ce cObject assez simple trace une ligne horizontale pour laquelle vous spécifiez l'épaisseur et la couleur, ainsi que l'espacement à gauche et à droite.

Exemple :

```

10 = HRULER
10 {
    lineThickness = 1
    spaceLeft = 20
}

```

IMGTEXT

Le cObject IMGTEXT aide à agencer les images et le texte. Il est utilisé d'habitude pour générer l'élément de contenu **Text w/Image**.

Les images sont placées dans un tableau placé avant ou après le texte et à sa gauche ou à sa droite.

Exemple : le objet IMGTEXT de styles.content (default) est complètement remplacé.

```
temp.imagetext = IMGTEXT
temp.imagetext {
  text < tt_content.text.20
  imgList.field = image
  textPos.field = imageorient
  imgPath = uploads/pics/
  imgObjNum = 1
  1.file.import.current = 1
  maxW = 150
  border = 1
  textMargin = 10
}

tt_content.textpic.20 >
tt_content.textpic.20 < temp.imagetext
```

CASE

CASE permet de faire la distinction entre différents cas, à la manière de la construction switch du PHP. Si la propriété key a le même nom qu'une autre propriété du tableau de l'objet, alors ce dernier est affiché.

L'appellation du cObject peut être choisie librement mais les mots réservés **key**, **default**, **stdWrap** et **if** ne peuvent pas être utilisés.

Si la propriété **key** n'est pas définie, alors l'objet **default** est utilisé à sa place. Exemple : pour le type de contenu **Text**, la disposition 2 est affichée avec un en-tête de section 2 en minuscules.

```
temp.stuff = CASE
temp.stuff.key.field = CType
temp.stuff.default < lib.stdheader
temp.stuff.text < lib.stdheader
lib.stdheader >
lib.stdheader < temp.stuff
lib.stdheader.text {
  10.2.case = lower
}
```

LOAD_REGISTER

Le registre est une mémoire interne temporaire en TYPO3. Les variables stockées sont globalement disponibles et sont affichées, sauveées ou écrasées à différents moments de l'analyse syntaxique. Avec **LOAD_REGISTER**, les valeurs sauveées temporairement peuvent être écrasées par le système et remplacées par vos propres définitions, de manière à influencer le processus de génération du contenu. La mémoire temporaire est organisée en tableau et comprend des propriétés et leurs valeurs.

Exemple : la date à laquelle a été réalisée la dernière mise à jour de la page est remplacé par une date fixée.

```
10 = LOAD_REGISTER
10.SYS_LASTCHANGED.data = date: U
```

RESTORE_REGISTER

Cet objet annule les changements réalisés par `LOAD_REGISTER` dans le registre interne du système interne. Le tableau de registre original est alors restauré. Si les changements de registre doivent avoir un effet sur un endroit précis, et doivent ensuite être à nouveau remplacés par les valeurs du système, alors l'utilisation de `RESTORE_REGISTER` est appropriée.

Exemple :

```
20 = RESTORE_REGISTER
```

FORM

L'objet `FORM` est responsable de la représentation des formulaires. Il permet de définir des formulaires directement en TypoScript (par exemple les formulaires de messagerie et d'identification du système) et permet aussi à des rédacteurs de les définir dans le backend du système.

Exemple : formulaire d'identification.

```
temp.loginform = FORM
temp.loginform {
    dataArray {
        10.label = Nom d'utilisateur:
        10.type = *user=input
        20.label = Mot de passe:
        20.type = *pass=password
        30.type = logintype=hidden
        30.value = Login
        40.type = submit=submit
        40.value = Identification
    }
    layout = <tr><td>###LABEL###</td><td>###FIELD###</td></tr>
    stdWrap.wrap = <table>|</table>
    hiddenFields.pid = TEXT
    hiddenFields.pid.value = 123
}
```

SEARCHRESULT

Le cObject `SEARCHRESULT` contrôle et liste le résultat de la recherche TYPO3. La recherche se fait uniquement pour les types de pages **Standard**, **Avancé** et **Hors menu**. Le cObject détermine l'apparence de la liste en plus des valeurs des tables et des champs à inclure dans la recherche. Les termes de la recherche sont fournis par un formulaire avec des paramètres URL spéciaux. Les requêtes sur la base de données se font en arrière-plan et la liste des résultats est renvoyée. Les termes de la recherche sont conservés dans le registre décrit ci-dessus pour que les termes trouvés puissent par exemple être mis en couleurs par d'autres fonctions.

USER et USER_INT

Ces objets de contenu sont les instruments recommandés pour l'intégration de vos propres scripts PHP. L'objet appelle une fonction PHP ou une méthode d'une classe et leur configuration peut être passée en argument. Presque tous les plugins du frontend sont basés sur cet objet.

L'objet `USER_INT` est surtout utilisé lorsque sont impliquées des fonctionnalités dynamiques qui ne peuvent ou ne devraient pas être dans le cache. Par exemple, si l'utilisateur soumet un formulaire d'inscription à un bulletin d'information, cette donnée, pour être sauveée dans la base de données, sera traitée dynamiquement.

Exemple :

```
includeLibs.alt_xml = media/scripts/xmlversionLib.inc
10 = USER
10.userFunc = user_xmlversion->main_xmlversion
```

PHP_SCRIPT

`PHP_SCRIPT` sert à intégrer de simples scripts PHP. Un des désavantages de cet objet est que vous devez désactiver complètement la fonctionnalité de cache de la page, ce qui entraîne des délais perceptibles lors de l'appel de la page et une charge accrue sur le serveur.

Pour intégrer vos propres scripts, vous devriez autant que possible utiliser `USER` ou `USER_INT`. Cette option est ignorée si

```
$TYPO3_CONF_VARS["FE"]["noPHPscriptInclude"] = 1;
```

est activé et placé dans `localconf.php`.

PHP_SCRIPT_INT

Cette alternative à `PHP_SCRIPT` améliore quelque peu les problèmes de cache mais cause des problèmes de débogage et de variables globales. Utilisez plutôt `USER` ou `USER_INT`.

PHP_SCRIPT_EXT

Avec cet objet, l'intégration des scripts se déroule hors de l'environnement TypoScript. Ceci est très avantageux, mais le traitement se passe en grande partie « en dehors » de TYPO3. Vous devriez donc utiliser `USER` ou `USER_INT`.

TEMPLATE

Ce cObject vous permet de charger un gabarit HTML. Par contraste avec `FILE`, vous pouvez limiter le contenu du fichier HTML à certaines zones précises. Vous pouvez aussi spécifier quel contenu remplacera quel marqueur.

Exemple :

```
page.10 = TEMPLATE
page.10 {
    template = FILE
    template.file = fileadmin/templates/test.tmpl
    workOnSubpart = DOCUMENT
    subparts {
        HELLO = TEXT
        HELLO.value = Ce texte remplace la sous-partie "HELLO"
    }
    marks {
        TESTMARKER = TEXT
        TESTMARKER.value = Ce texte remplace le marqueur "TESTMARKER"
    }
}
```

MULTIMEDIA

Cet objet permet d'intégrer différents fichiers multimédias. Quelques types de fichiers possibles sont : TXT, HTML, HTM, CLASS, SWF, SWA, DCR, WAV, AU, AVI, MOV, ASF, MPG et WMV.

Selon le type de fichier, vous pouvez spécifier certains paramètres du fichier.

Exemple :

```
10 = MULTIMEDIA
10.file = fileadmin/video/test.avi
10.params.width = 200px
```

EDITPANEL

Dans le contexte du concept d'« édition de frontend », les options d'édition d'une page sont rendues disponibles dans le frontend pour des utilisateurs identifiés dans le backend.

Exemple :

```
temp.editPanel = COA
temp.editPanel {
  10 = EDITPANEL
  10 {
    allow = toolbar,hide,move,delete
    label = Page courante:<B>%s</B><br /> vous pouvez créer ici une
           sous-page ici<br />,
           éditer ou déplacer la page
    line = 5
  }
  20 = EDITPANEL
  20 {
    newRecordFromTable = tt_news
    line = 0
    label = Nouveau contenu<br /> Vous insérer ici une actualité.
  }
}
```

5.3.5 Objets de premier niveau

Pour restituer du contenu sur une page, on a utilisé jusqu'à présent l'objet `page` assigné au type d'objet `PAGE`. Ici, l'objet `page` représente un *objet de premier niveau* (TLO pour *TopLevel/Object*). Il est situé au plus haut niveau de la hiérarchie des objets, ou, d'une autre façon, on pourrait dire qu'il est à l'extrémité gauche de la configuration TypoScript. TypoScript utilise une série de TLO qui influencent l'application. Dans la zone « setup » de TSref sont repris les TLO déjà définis ainsi que la liste des propriétés et objets qu'ils intègrent (par exemple `FE_TABLE`, `FRAMESET`, `FRAME`, `META`, `CARRAY`). Plusieurs objets de premier niveau tels que `config`, `constants` et `FEData` ne doivent pas être d'abord assignés à une type de données pour être définis, étant donné qu'ils sont déjà prédéfinis.

Référence 110842

types

Type de donnée : `readonly`

Réservé à l'usage interne (`class.t3lib_tstemplate.php`) ; par exemple `type=99` est réservé à l'affichage de texte en clair.

resources

Type de donnée : `readonly`

Utilisé en interne pour enregistrer dans une liste séparée par des virgules les ressources pour la hiérarchie des gabarits.

sitetitle

Type de donnée : `readonly`

Réservé à l'usage interne pour insérer le titre du site Web à partir des gabarits.

config

Type de donnée : `->CONFIG`

config permet de définir les valeurs de configuration globale pour l'édition des pages, le cache, le débogage, la manipulation des liens, etc. Ces valeurs sont sauveées pour les pages cachées, ce qui signifie qu'elles sont aussi disponibles si une page est appelée à partir du cache.

constants

Type de donnée : `->CONSTANTS`

Définit les constantes qui remplacent dans tout le site les passages de texte marqués dans les éléments de contenu. Ceci est réalisé lorsque les éléments de contenu sont passés à la fonction `parseFunc`. Exemple :

```
constants.WEBMASTER_EMAIL = webmaster@mon-domaine.com
```

FEData

Type de donnée : `->FE_DATA`

FEData configure le traitement des données GET et POST envoyées par les utilisateurs via le frontend. Il s'agit d'un concept ancien qui est désormais à peine utilisé par les extensions. Exemple : le livre d'or.

includeLibs

Type de donnée : `tableau de chaînes de caractères`

Vous permet d'intégrer des fichiers PHP qui ont leurs propres bibliothèques de fonctions à TYPO3, de sorte que ces fonctions soient accessibles. Notez que l'objet **PAGE** fournit des propriétés avec la même fonctionnalité (`page.includeLibs`). Mais ces propriétés sont insérées après le TLO `includeLibs` et peuvent écraser des valeurs dans le TLO. Exemple :

```
includeLibs.gmenu_foldout = media/scripts/gmenu_foldout.php
```

plugin

Les plugins des extensions sont insérés dans le TLO **plugin**. Les plugins chargés peuvent être vus comme des sous-objets de **plugin** via leur clé d'extension.

tt_*

Le TLO `tt_*` est utile pour restituer du contenu des tables. Le gabarit standard `content (default)` normalement utilisé contient une définition correspondante de la table `tt_content`, et contrôle donc, par-là même, la restitution de la plupart des éléments de contenu.

temp

Les TLO **temp** et **styles** sont utilisés comme des bibliothèques de code que vous pouvez copier pendant l'analyse syntaxique. Ils ne sont pas stockés dans le cache avec le gabarit mais ils sont toutefois supprimés/vidés avant que le gabarit ne soit caché ! Utilisez **temp** pour enregistrer vos propres données temporaires.

styles

Se comporte de la même manière que **temp**, mais n'est pas disponible pour votre usage, mais bien pour celui des gabarits par défaut.

lib

Sert à définir des bibliothèques de code auxquelles des références sont faites.

...

Type de donnée : **PAGE**

Vous pouvez choisir le titre que vous désirez. Mais nous vous recommandons d'utiliser le titre **page** pour la définition du TLO de type **PAGE**.

...

Type de donnée : n'importe quel type

Définissez vos propres TLO et assignez-les aux types de données correspondants.

Les TLO utilisés et la hiérarchie des objets sont affichés clairement dans le TypeScript Object Browser :



Figure 5.33:
Les objets de premier
niveau du gabarit
affichés dans le
TypeScript Object
Browser

Exemples**config**

Certaines fonctions sont configurées globalement avec **config**. Par exemple si vous fixez **config.simulateStaticDocuments=1** dans le champ **Setup** du gabarit TypeScript, le résultat dans le frontend créé dynamiquement est simulé comme s'il était composé de pages statiques⁵.

Avec **config.cache_period=432000**, vous fixez le cache des pages à cinq jours.

⁵Pour les spécifications, voir TSref

```

config {
  simulateStaticDocuments = 1
  cache_period = 432000
}

```

constants

Avec le TLO **constants**, vous définissez des constantes pour l'application dans son ensemble. Ceci vous permet de remplacer les balises spécifiques par le texte correspondant, en utilisant la fonction **parseFunc** et sa propriété **constants=1**. Si l'analyseur syntaxique trouve dans le contenu de l'application la constante de l'exemple **###WEBMASTER_EMAIL###**, alors le texte **webmaster@mon-domaine.com** est affiché sous forme de lien.

```

constants {
  WEBMASTER_EMAIL = <a href="mailto:webmaster@mon-domaine.com">web
master@mon-domaine.com</a>
}

```

lib

Le TLO **lib** est prévu pour les bibliothèques de code réutilisable qui peuvent être référencées. Un texte comme celui de l'exemple n'est bien sûr pas vraiment adéquat !

```

lib.welcomingText = TEXT
lib.welcomingText {
  value = TypeScript est simplement génial.
  wrap = <p> | </p>
}

page = PAGE
page {
  typeNum = 0
  10 = TEXT
  10.value = Hello Typos!
  20 = < lib.welcomingText
}

```

temp et styles

Référence 110841

Les objets de premier niveau **temp** et **styles** sont particuliers en ceci qu'ils sont supprimés du champ **Setup** après qu'ils ont été analysés par TypeScript. Cela signifie qu'il est impossible de faire référence à l'objet **temp.monObject** puisque **temp** n'existe plus pendant le traitement du champ **Setup**.

De fait, pour améliorer les performances, toutes les valeurs TypeScript sont généralement enregistrées dans la table de base de données **cache_hash** après qu'elles ont été lues. Si certaines parties de TypeScript doivent être stockées temporairement, utilisez les objets de premier niveau **temp** et **styles**. Leurs objets, leurs propriétés et leurs valeurs sont alors copiés en différents endroits de TypeScript et sont réutilisables. Après l'analyse syntaxique, toutefois, ils sont supprimés, ce qui permet d'alléger les gabarits stockés dans le cache. L'objet de premier niveau **temp** est destiné à vos objets généraux, alors que **styles** est réservé aux gabarits standards (gabarits statiques) inclus dans TYPO3.

5.4 Outils de développement

TYPO3 fournit certains outils permettant au développeur de programmer plus facilement en TypoScript. D'une part le module **Web** → **Gabarit** dans le backend propose de nombreuses fonctions, et d'autre part, le **panneau d'administration** peut être affiché dans le frontend pour fournir de l'aide supplémentaire.

5.4.1 Info/Modify

Le module **Web** → **Gabarit** → **Info/Modify** vous montre la liste des gabarits existant dans l'arborescence des pages. Pour cela, choisissez la page racine comme page courante en cliquant sur le titre du globe dans l'arborescence des pages. Cette liste indique les pages et leurs gabarits et distingue parmi ceux-ci les gabarits racines des gabarits d'extension.

Le module aide également à configurer de nouveaux gabarits lorsque la page sélectionnée n'a pas de gabarit qui lui est associé. Si nécessaire, vous pouvez choisir un gabarit statique.

Vous avez en outre la possibilité de sauter au gabarit le plus proche de la page considérée.

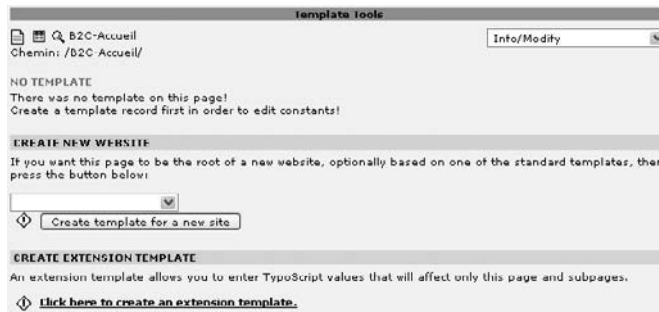


Figure 5.34:
Info/Modify permet
de créer de nouveaux
gabarits

Si vous avez choisi une page associée à un gabarit, vous pouvez éditer ce dernier en sélectionnant, soit des champs précis, soit l'enregistrement complet. Si vous éditez un champ, l'assistant TS est disponible (voir à la section suivante). De plus, le cache est supprimé automatiquement.

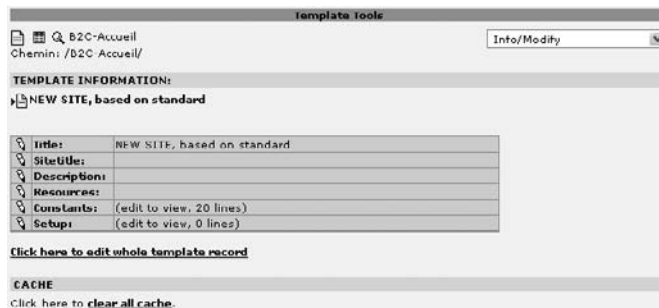


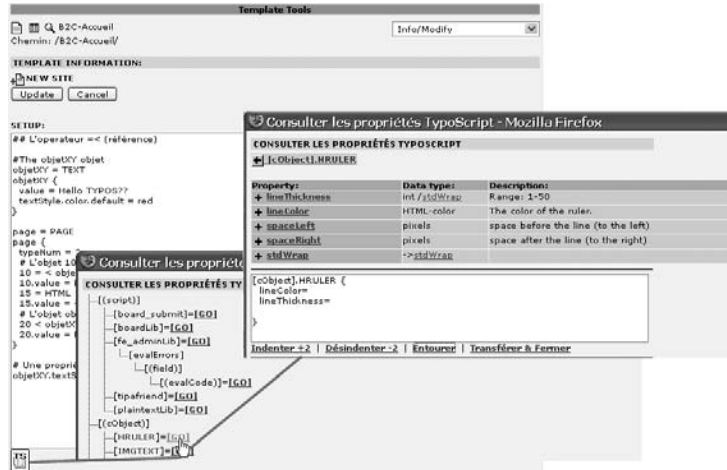
Figure 5.35:
Édition des
enregistrements de
gabarits existants

5.4.2 Assistant TS

De même que l'édition de contenu pour des types de contenu comme **Tableau** ou **Formulaire** se fait à l'aide d'un assistant, il en existe aussi pour TypeScript. Vous pouvez naviguer dans les types d'objets et les propriétés et les sélectionner.

À la figure suivante, vous voyez comment ouvrir un champ **Setup TS** pour l'éditer, avec le module **Web** → **Gabarit** → **Info/Modify**. Avec l'icône assistant TS, vous ouvrez l'assistant dans une nouvelle fenêtre.

Figure 5.36:
Choix d'objets
TypeScript et de leurs
propriétés avec
l'assistant



Les types d'objets sont d'abord affichés, chacun d'eux s'ouvrant à l'aide du lien **[GO]** permettant une vue détaillée. Dans l'exemple, c'est le cObject **HRULER** qui est ouvert. La vue détaillée montre les propriétés du type d'objet ainsi que ses types de données, sa description et ses valeurs par défaut. Si vous cliquez sur le nom d'une propriété, l'assistant se ferme et la propriété est insérée dans le formulaire d'entrée. Dans notre exemple, les propriétés voulues ont d'abord été collectées dans le champ d'entrée de l'assistant par un clic sur l'icône « + ». Ensuite, les propriétés ont été entourées du type d'objet en utilisant **Entourer**. Enfin, lorsque vous cliquez sur le lien **Transférer & Fermer**, le code est transféré.

Référence 090547

Même si l'assistant ne peut remplacer complètement TSref, il contient tout de même la majorité de l'information. Les données affichées sont créées directement à partir du document TSref-OpenOffice. Même les propriétés des extensions sont lues à partir de la documentation correspondante. Découvrez la manière de préparer votre documentation d'extension à partir de la référence ci-contre.

5.4.3 TypeScript Object Browser

Le TypeScript Object Browser (**Web** → **Gabarit** → **Object Browser**) a déjà été utilisé quelquefois dans les sections précédentes pour afficher le champ **Setup**. Outre l'affichage des champs **Setup** et **Constants**, l'Object Browser fournit également des fonctions pour tester les conditions et pour naviguer entre les constantes et les afficher dans le setup. En plus, l'Object Browser permet d'éditer les propriétés.

L'Object Browser fonctionne toujours sur base de la page courante ou du gabarit associé. Toutefois, en plus des données du gabarit sélectionné, toutes les données de gabarit accumulées depuis la page racine sont montrées. Le TypeScript affiché est celui qui est actif dans le front-end lorsque la page associée est montrée. Pour tester le comportement des conditions définies dans le gabarit, vous pouvez les activer individuellement.



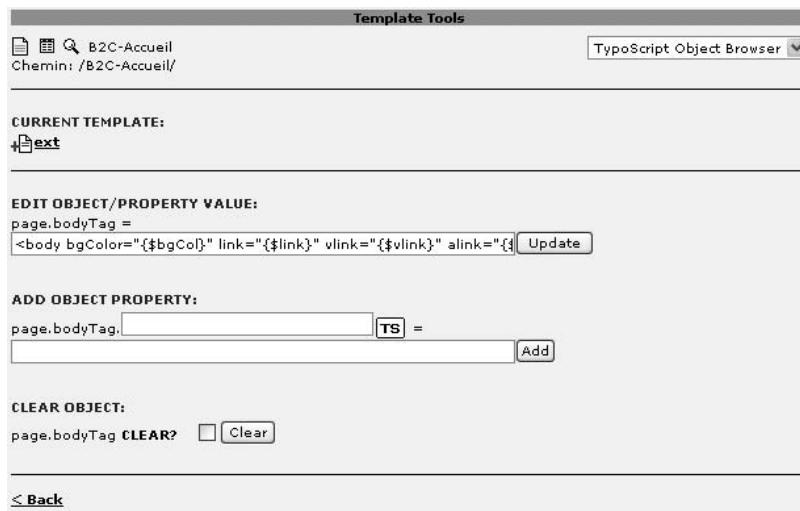
Figure 5.37:
TypoScript Object
Browser dans le
module Web →
Gabarit

La fonction de recherche passe en revue les valeurs de propriétés et affiche les résultats en couleurs. Si vous choisissez l'option **Use ereg()**, vous pouvez utiliser des expressions rationnelles.

Comme indiqué à la figure précédente, vous pouvez mettre en évidence les constantes dans le champ **Setup** en choisissant d'afficher soit les constantes originales, soit les valeurs de remplacement.

Si vous cliquez sur le nom d'une partie d'un chemin (par exemple **property**), vous obtenez le formulaire suivant :

Figure 5.38:
Édition d'une
propriété dans
l'Object Browser



Ce faisant, vous avez l'occasion d'éditer la valeur, d'ajouter une propriété ou de supprimer l'élément choisi avec >.

Le code TypeScript modifié ou ajouté est écrit dans le gabarit actuel en s'ajoutant à la fin du code existant. Il peut donc arriver après un certain temps qu'une masse de code s'accumule et que vous deviez la trier à la main. Mais cette option d'édition est d'une grande aide, d'autant que l'assistant TS y est également disponible.

5.4.4 Template Analyzer

Le module **Web** → **Gabarit** → **Template Analyzer** a déjà été mentionné dans le contexte des gabarits en cascade, puisqu'il permet entre autres d'afficher la hiérarchie du gabarit actif.

Le Template Analyzer montre la hiérarchie logique du gabarit pour la page sélectionnée – les gabarits utilisés pour la restitution dans le frontend.

Le tableau indique en outre si les options **Rootlevel**, **Clear Constants** et **Clear Setup** sont activées pour le gabarit – et indique aussi l'ID de la page et le niveau (Rootlevel) auquel est situé le gabarit. La dernière colonne montre les ID des gabarits intégrés avec **Template on next level**.

À la figure suivante, **Clear Constants** a été activé pour le gabarit appelé « An Extension Template » ; ce faisant, les constantes définies dans les gabarits précédents sont désactivées.

Si vous cliquez sur le titre d'un gabarit, son code s'affiche. Dans l'exemple, nous voyons le code du gabarit « Main Template », mais pas les constantes, bien que certaines soient définies dans ce gabarit. Cela s'explique par le fait que le gabarit « An Extension Template » est actif sur cette page, avec **Clear Constants** sélectionné. Les constantes sont donc désactivées et ne sont pas affichées.

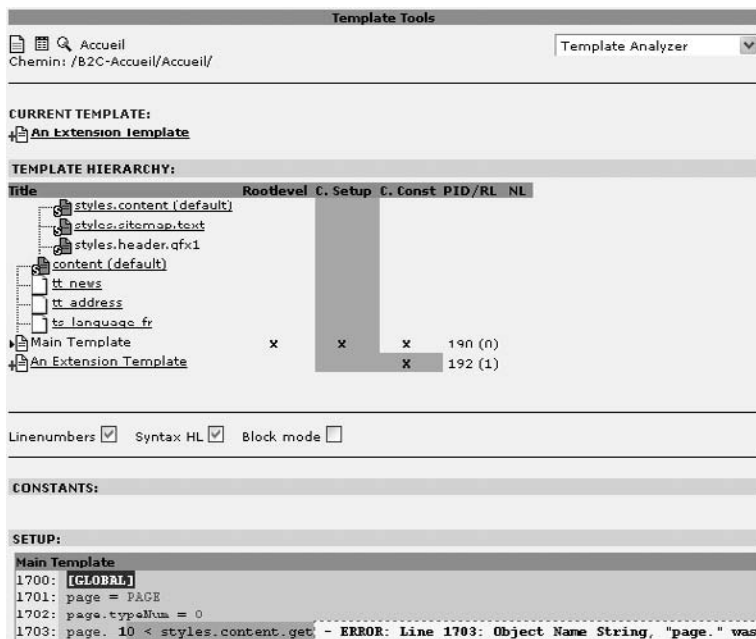


Figure 5.39:
Le Template Analyzer
montre la hiérarchie
et le code de certains
gabarits

Vérifier la syntaxe

Vous ne pouvez pas tester TypoScript pour vérifier si un objet non-existant est appelé ou si des propriétés ou des valeurs ont été mal nommées. Ce sont souvent des petites fautes de frappe qui nous empêchent d'arriver au résultat voulu. Vous pouvez seulement vérifier si la syntaxe du code est correcte.

À partir de la version 3.6.0 de TYPO3, l'option **Syntax highlighting** est disponible aux développeurs sous **Template Analyzer**.

Dans l'exemple ci-dessus, un espace a été ajouté au chemin d'objet. L'analyse syntaxique vous indique dès lors que cet espace n'est pas suivi d'un opérateur.

5.4.5 Constant Editor

Comme nous l'avons décrit à la section 5.2.8, les constantes servent, dans un enregistrement de gabarit, à faire passer des valeurs globales ou faciles à manipuler au code du champ **Setup**. Vous pouvez éditer ces constantes de manière simple et claire à l'aide du **Constant Editor** (**Web** → **Gabarit** → **Constant Editor**). De cette façon, grâce aux gabarits standards, qui configurent des pages Web, changer l'apparence et la fonctionnalité du site avec des constantes devient très simple.

Les gabarits standards sont particulièrement faciles à configurer avec le Constant Editor. Une capture d'écran du gabarit et sa description sont montrés à la figure suivante.

Figure 5.40:
Édition du gabarit
standard GREEN avec
le Constant Editor



Le Constant Editor renvoie les valeurs préconfigurées (par défaut) pour le gabarit choisi. Si vous devez remplacer certaines valeurs, activez les cases à cocher concernant les constantes et rechargez le formulaire. Elles sont ainsi placées dans les champs du formulaire et sauveées dans le champ Constants de l'enregistrement de gabarit sélectionné.

Mise en évidence des constantes pour le Constant Editor

Si vous voulez mettre à disposition vos constantes dans le Constant Editor, vous devez posséder de l'information supplémentaire, entre autres sur les descriptions et les intervalles des valeurs admises.

Dans ce but, les définitions des constantes doivent être précédées par des commentaires contenant des règles syntaxiques précises.

Exemple :

```
# cat=temp.monTexte; type=string; label=Mon Texte
monTexte.monTexteInput = Hello World!
```

Contrairement à TYPO3, le Constant Editor tient compte des commentaires lors de l'analyse syntaxique des gabarits. Il groupe ces commentaires par catégories sur base des données additionnelles dans les commentaires spéciaux, et il les affiche avec un descriptif (label) dépendant de leur type.

Référence 969386

Règles syntaxiques pour les commentaires :

- Le commentaire est placé à la ligne précédant la constante donnée.
- Chaque ligne de commentaire est divisée en paramètres séparés par un point-virgule.
- Chaque paramètre consiste en une clé et une valeur, séparées par un signe égal. TYPO3 fournit les clés suivantes : **cat** (catégorie et éventuellement sous-catégorie), **type** (type de constante) et **label** (texte explicatif).

Le gabarit **mon texte** suivant sert d'exemple. Le gabarit définit le texte et sa présentation à l'aide de constantes.

Données dans le champ **constants** du gabarit **mon texte** :

```
# cat=montexte/ctext/a; type=string; label=Mon Texte
monTexteInput = Hello World!
# cat=montexte/ctext/b; type=typo; label=Ma Police
maPoliceTexte = Verdana
# cat=montexte/ctext/d; type=color; label=Ma Couleur
maCouleurTexte = red
# cat=montexte/ctext/c; type=small; label=Ma taille de Texte
maTailleTexte = 32
```

Données dans le champ **setup** du gabarit **mon texte** :

```
temp.monContenu = TEXT
temp.monContenu {
    value = $monTexteInput
    textStyle.face.default = $maPoliceTexte
    textStyle.size.default = $maTailleTexte
    textStyle.color.default = $maCouleurTexte
}
```

Les utilisateurs peuvent facilement modifier les constantes dans le Constant Editor. La figure suivante indique les valeurs courantes du gabarit **mon texte** et de la catégorie **montexte**.

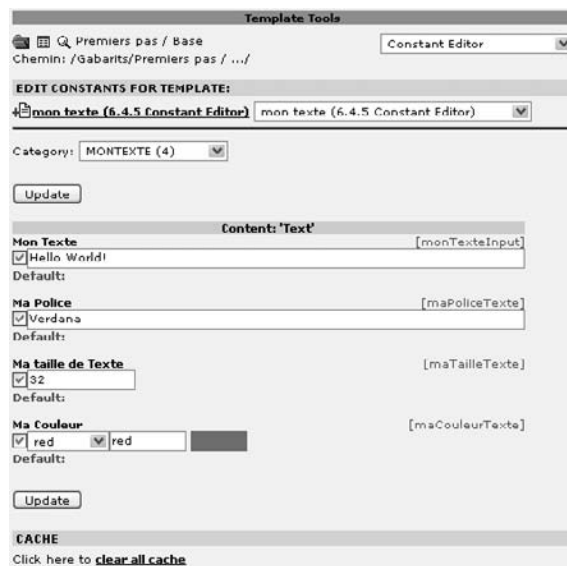


Figure 5.41:
Constantes du
gabarit **mon texte**
dans le Constant
Editor

Paramètres

Catégories (cat)

Vous pouvez choisir vos catégories librement et elles peuvent rassembler différentes constantes. C'est pourquoi il est judicieux de choisir des termes généraux pour désigner ces groupements. Il est également possible d'utiliser les catégories préconfigurées qui suivent pour les gabarits standards.

basic

La catégorie **basic** rassemble les constantes importantes pour les gabarits de mise en forme. Ces constantes sont celles de base que vous configurerez probablement toujours ; elles contiennent entre autres la taille maximum des images et permettent d'activer plusieurs propriétés.

menu

Les réglages du menu dépendent du type ; ils comprennent les fichiers de police de caractère, les tailles et les arrière-plans.

content

Les constantes qui influencent l'affichage des éléments de contenu.

page

Configurations générales concernant la page telles que des balises méta, des destinations de liens, etc.

advanced

Fonctions avancées rarement utilisées.

plugin.*

Constantes pour certains plugins (extensions).

Exemple : **plugin.meta**

Sous-catégories

Les sous-catégories regroupent les constantes d'une catégorie dans une zone du Constant Editor et sont ajoutées à une catégorie en utilisant le symbole /. Seules les sous-catégories définies dans la liste ci-dessous sont disponibles. Toutes les autres constantes sont automatiquement reprises dans la sous-catégorie **Others**. Spécifiez l'ordre dans lequel les constantes sont classées en ajoutant une lettre de a à z. Si vous ne le faites pas, la lettre z est prise comme valeur par défaut.

Exemple :

```
# cat=basic/enable/b; type=string; label= Ma Constante
```

Sous-catégories existantes :

enable

Pour les constantes servant à activer ou désactiver les fonctions de base du gabarit.

dims

Pour les dimensions en tout genre telles que les pixels, les largeurs et hauteurs d'images, les cadres, etc.

file

Constantes définissant des fichiers comme les images en arrière-plan, les polices de caractère, etc. ; d'autres options concernant le fichier sont également spécifiées.

typo

Pour la typographie et ses constantes.

color

Aide à rassembler les définitions des couleurs ; toutefois, de nombreuses couleurs et leurs options sont aussi incluses dans d'autres catégories.

links

Pour utiliser avec des liens ; typiquement, **target**.

language

Options spécifiques à la langue.

cheader, **cheader_g**, **cctx**, **cimage**, **cbullets**, **ctable**, **cuploads**,
cmultimedia, **cmailform**, **csearch**, **clogin**, **csplash**, **cmenu**, **cshortcut**, **clist**,
cscript, **chtm**

Ces sous-catégories sont basées sur les éléments de contenu prédéfinis de TYPO3 tels que du texte, un en-tête, une image, etc.

Type de champ (type)

type vous permet de spécifier le type de champ et de définir les options d'édition pour l'interface utilisateur du Constant Editor.

int [low-high]

Un nombre entier ; l'intervalle des valeurs admises peut être spécifié.

Exemple : **int [0-10]**

int+

Nombre entier positif.

color

Code couleurs HTML.

wrap

Permet d'éditer une enveloppe.

offset [L1,L2,...,L6]

Liste de nombres entiers séparés par des virgules ; vous pouvez spécifier jusqu'à six paramètres entre crochets. Les paramètres par défaut sont **x,y**.

options [item1,item2, ...]

Champ de sélection avec titres/valeurs, dans lequel les éléments sont séparés par des virgules ; titre et valeur sont séparés par **=**.

Exemple : **options [titre1=valeur1, titre2=valeur2]**

boolean [valeur vraie]

Booléen ; vous pouvez spécifier éventuellement la valeur de « true ». La valeur par défaut est 1.

comment

Booléen ; sélectionné = « », non-sélectionné = « # ».

Ce type sert à commenter du texte, et donc à désactiver du code TypeScript à l'aide d'une constante.

file [ext-list /IMAGE_EXT]

Sert à sélectionner les ressources des fichiers ; il est possible d'envoyer directement des fichiers vers le serveur pour qu'ils soient associés au champ Ressources de l'enregistrement gabarit. Vous pouvez restreindre cette sélection à certains types de données précis. Pour ce faire, une liste de types de données autorisés est spécifiée dans la liste, par exemple [ttf] ou [txt,html,htm] (sans espace). Vous pouvez aussi entrer IMAGE_EXT. Dans ce cas les types de fichier image par défaut sont permis.

string

Champ d'entrée de texte.

En-tête et description (label)

À l'aide du type label vous pouvez spécifier un en-tête et un texte descriptif, séparé par « : » :

```
# cat=montexte/ctext/a; type=string; label= Texte: Le texte apparaît sur la droite ...
monTexteInput = Hello Wolrd!
```

Description des catégories (TSConstantEditor)

Vous pouvez aussi faire une description générale pour chaque catégorie de constantes. Cela s'avère utile, par exemple pour expliquer les gabarits standards à l'aide d'une capture d'écran et d'une description supplémentaire.

Les données additionnelles sont définies dans les constantes TLO TSConstantEditor. Cela vous permet d'assigner des icônes numériques à des constantes données et d'intégrer une image — généralement une capture d'écran.

Notez que le TSConstantEditor est défini dans le champ Constants du gabarit. L'utilisation des termes « TLO », « objet » et « propriété » dans le cas des constantes n'est pas correct strictement parlant puisque les constantes ne sont pas des objets. Nous les utiliserons néanmoins ici par facilité.

Exemple :

```
TSConstantEditor.montexte {
  header = Mon Texte
  description = Mon texte est une démo // il contient ...
  bulletlist = élément 1 // élément 2 // élément 3.
  image = gfx/scenario-1.png
  1 = monTexteInput, Texte
  2 = maPoliceTexte, Couleur
}
```

Nous résumons à présent les propriétés du TSConstantEditor :

header

Type de donnée : chaîne de caractères

En-tête de description ; affiché en majuscules.

description

Type de donnée : chaîne de caractères

Description du gabarit ; le passage à la ligne se fait avec la commande `//`.

bulletlist

Type de donnée : chaîne de caractères

Affiche une liste à puces ; les différents éléments sont séparés par deux barres obliques (`//`).

image

Type de donnée : image

Vous pouvez insérer une image pour illustrer les liens. Entrez les numéros des constantes décrites sur la figure et leur position dans le frontend. L'image doit être enregistrée dans le répertoire `gfx/` du module `Constant Editor (tstemplate_ceditor)` ou intégrée au champ des ressources de l'enregistrement de gabarit.

Array, 1-20

Liste de noms de constantes

Les icônes numérotées en rouge sont assignées aux constantes reprises dans le Constant Editor via la liste numérique. Elles font le lien avec l'image spécifiée.

5.4.6 Panneau d'Administration

Le Panneau d'Administration fournit des fonctions aux éditeurs, mais fournit aussi de l'information au développeur TypeScript. Pendant la restitution du contenu, TSFE retrace le processus de restitution ; vous pouvez afficher les informations de débogage dans le panneau d'administration.

Si le panneau d'administration est configuré conformément à la section ⁶, les options suivantes sont disponibles dans la zone **TypoScript** :

Afficher l'arborescence

Lorsque cette option est activée, le résultat est affiché sous forme d'arbre.

Afficher toutes les dates

Cette option donne le temps nécessaire à la restitution des éléments de la page, en millisecondes.

Afficher les messages

Des explications supplémentaires et des messages d'erreur sont indiqués dans le log de la restitution, que vous activez avec cette option.

Suivre la génération du rendu

Cette option donne de l'information sur la restitution des éléments de contenu. Le résultat est restreint à de l'information générale et à des objets PAGE.

Afficher le contenu

Lorsque cette option est active, vous pouvez afficher le contenu généré. Ceci vous permet de voir le code HTML pour l'objet TS correspondant.

⁶La configuration du panneau d'administration est décrite à la section 4.7

Export t3d

L'export des enregistrements se fait à l'aide du menu contextuel **Plus d'options...** → **Exporter vers un .t3d**. D'abord, vous sélectionnez dans l'arborescence la page à partir de laquelle l'export est initié.

Le formulaire suivant sert à spécifier la portée de l'export qui peut s'étendre de simples pages jusqu'à des arborescences des pages complètes.

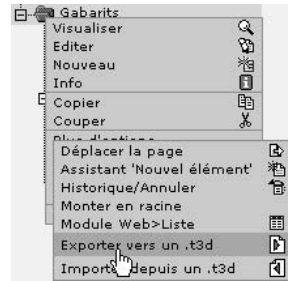


Figure 5.43:
Appel de l'export t3d
à partir du menu
contextuel

Configurez l'export dans la partie supérieure du formulaire. Différentes propriétés et fonctionnalités sont disponibles.

Figure 5.44:
Détermination des
propriétés d'export

ID de page

ID de la page à partir de laquelle l'export est réalisé.

Arborescence

L'arborescence à exporter, sélectionnée via la page d'initiation de l'export et la profondeur des niveaux.

Niveaux

Spécifiez la profondeur des niveaux à exporter dans le menu de sélection (commencez par la page de démarrage sélectionnée).

Include tables

Vous permet de décider du contenu des tables de bases de données à inclure dans l'export, contenu inclus dans la section de l'arborescence des pages considérées.

Include relations to tables

Si des enregistrements sont liés à des enregistrements situés hors de l'arborescence des pages choisies, définissez ici les tables à inclure.

Use static relations for tables

Sélectionnez les tables en plus des tables statiques pour lesquelles il ne faut pas changer les relations. Ce procédé a priorité sur les tables reprises dans la section précédente, « Include relations to tables ». C'est utile par exemple pour les tables « sys_languages » ou les structures TemplaVoilà, où le système de destination de l'import peut contenir les mêmes enregistrements pour ces tables alors que vous importez librement les pages et le contenu.

Exclude elements

Reprend ici la liste des enregistrements que vous avez exclus de l'import dans la liste reprise dans la section **Structure à exporter**.

Grâce au bouton **Update**, les configurations choisies sont appliquées. La structure des enregistrements choisis pour l'export est exposée en détail dans la zone inférieure. Si des relations sont perdues, elles sont marquées par **Lost relation**. Si les détails sont corrects, l'export démarre avec le bouton **Download export**⁷ dans la section **File & Preset** du formulaire.

Import t3d

L'import de données via un document TYPO3 est lui aussi lancé à partir du menu contextuel à l'aide du menu **Plus d'options...** → **Importer depuis un .t3d**. Le point de départ est la page dans l'arborescence des pages à partir de laquelle les données sont intégrées. Puisque l'import via la page racine (globe) est impossible, vous pouvez démarrer un import via n'importe quelle page, par exemple, et bouger ensuite l'arborescence des pages importées.

⁷Si vous rencontrez des difficultés à télécharger, il faut savoir qu'un bogue dans Internet Explorer 5.5 tente de sauver la page dans le backend comme un téléchargement.

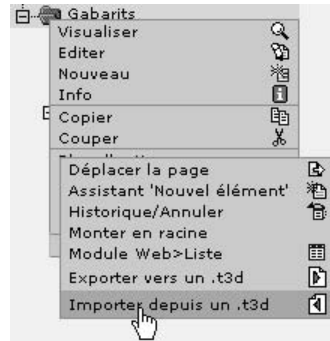


Figure 5.45:
Lancement de
l'import t3d via le
menu contextuel

Le document TYPO3 (.t3d), pour être importé, doit se situer dans le répertoire fileadmin/. Vous pouvez transférer directement un fichier vers l'onglet **Envoyer**, de sorte qu'il soit sélectionnable pour l'import via le menu de sélection. Le bouton **Prévisualiser** montre une prévisualisation des enregistrements à importer.

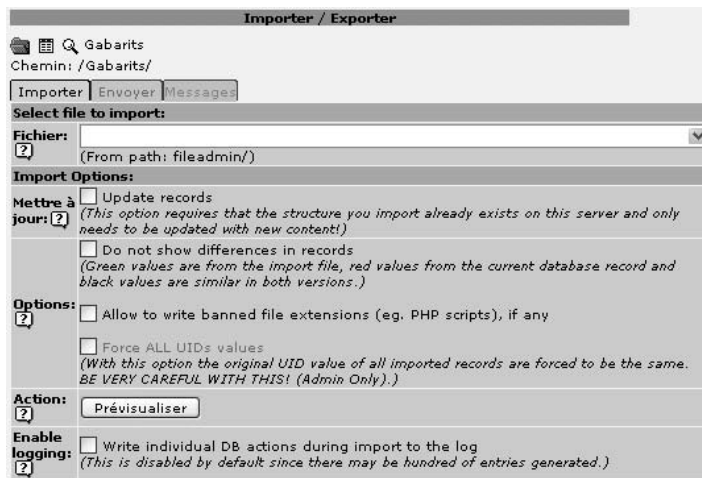


Figure 5.46:
Interface d'import
pour un document
TYPO3

5.5 Gabarits standards (gabarits statiques)

Tout paquetage TYPO3 contient déjà dans la table `static_template` des enregistrements qui sont seulement lisibles dans le backend ; mais ces enregistrements ne sont pas directement modifiables. Ces gabarits sont les ***gabarits standards*** bien connus et parfois appelés ***gabarits statiques***. Vous pouvez visualiser les gabarits standards dans une liste grâce au module **Web** → **Liste**, par exemple en sélectionnant la page racine (globe) dans l'arborescence des pages.

Les gabarits standards fournis forment une bibliothèque et sont utilisables pour vos propres sites Web. Ils constituent des gabarits pour le site complet. Ils contiennent également quelques

Référence 546011

composants TypeScript de base tels que content (default), qui définit l'affichage des éléments de contenu et leur mise en forme.

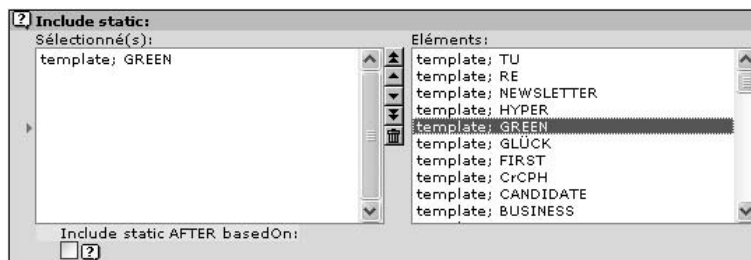
Figure 5.47:
Gabarits standards
contenus dans
l'installation basique
TYPO3 tels qu'ils sont
repris dans le module
Web → Liste



Un certain nombre de gabarits sont marqués de l'étiquette [DEPRECATED] qui signifie qu'ils sont considérés comme anciens ou dépassés ; ils sont inclus dans la liste pour des raisons de rétrocompatibilité. En particulier, la plupart des gabarits **plugin.*** sont devenus redondants depuis l'introduction du système d'extensions, puisque les gabarits correspondants sont fournis avec les extensions.

Pour utiliser les gabarits, insérez-les dans l'enregistrement de gabarit à l'aide du champ **Include static**.

Figure 5.48:
Insertion d'un gabarit
standard



Comme nous l'avons déjà mentionné, les extensions comprennent leurs propres gabarits. Ainsi, il est possible qu'elles offrent des gabarits standards que l'on peut sélectionner dans l'enregis-

trement de gabarit avec **Include static (from extensions)**. Ces gabarits assurent le même rôle que ceux disponibles dans **Include static**. De cette manière, même des gabarits définissant l'ensemble d'un site Web sont disponibles. Citons par exemple le gabarit **Green** implémenté comme une extension et que vous pouvez télécharger à partir du TER.

Les gabarits standards sont divisés en différentes catégories selon leur champ d'application et un préfixe correspondant leur est assigné (templates, plugin, contenu, ...). Pour continuer, voici une vue d'ensemble des différentes catégories.

5.5.1 content (default)

Ce gabarit standard est le plus utilisé. En général, il ne forme pas seulement la base du site Web et celle d'autres gabarits, mais il définit également le traitement et la représentation de base du contenu publié dans le frontend. Il applique les objets de contenu aux enregistrements correspondants de la table `tt_content`, selon le type de contenu spécifié par le rédacteur (le champ `CType` de la table `tt_content`). Ce gabarit se base sur `styles.content.default` qui définit en détail les différents types de contenu. De plus, `styles.sitemap.text` et `styles.header.gfx1` sont aussi inclus en tant que gabarits de base.

Le gabarit reste au centre de TYPO3, même si des développements récents ont gagné en importance. `content (default)` n'utilise pas seulement le TypoScript pour l'affichage, mais aussi en tant que logique de programmation — ce qui n'est en fait pas la tâche centrale du TypoScript. Mais les exigences envers le TypoScript ont changé, en particulier par l'adoption croissante du CSS pour la mise en page et par le besoin de favoriser l'accès à des sites Internet aux moins valides. Il est prévu que TYPOSCRIPT soit allégé en transférant certaines fonctionnalités au CSS et à des extensions. Ce concept est mis en pratique dans le gabarit `css_styled_content` dont l'extension porte le même nom.

De nombreux gabarits standards sont juste des composants de code intégrés dans un contexte plus large. Ces liens entre les gabarits sont visibles dans le Template Analyzer.



Figure 5.49:
content (default) et
les gabarits de base
intégrés dans le
Template Analyzer

5.5.2 styles.*

styles.content (default)
 styles.tmenu.pagetop
 styles.sitemap.text
 styles.sitemap.gs
 styles.img.logo
 styles.hmenu.tu (EXT1)
 styles.hmenu.tu
 styles.header.gfx1
 styles.gmenu_layer.green

styles.gmenu.first (EXT1)

styles.gmenu.first

styles.gmenu.bug

Cette catégorie de gabarit fournit des fragments de code à d'autres gabarits standards. Un grand nombre de ces gabarits est utilisé dans les gabarits de site (**template.***) qui se distinguent grâce à leur suffixe, par exemple **styles.hmenu.tu** et **styles.gmenu.bug**. D'autres, comme **styles.sitemap.text** et **styles.header.gfx1**, sont utilisés par le gabarit **content (default)**. Les composants ont été stockés dans des gabarits distincts pour que vous puissiez les réutiliser dans d'autres gabarits standards. Ils fournissent des objets de page préconfigurés tels que des menus, des plans de sites, des en-têtes et des logos.

styles.content (default)

Le gabarit **styles.content (default)** mérite une attention particulière ; il contient une sélection d'objets TypeScript prédéfinis à utiliser dans d'autres gabarits, en particulier pour la restitution de contenu de base dans le gabarit **content (default)**. L'élément le plus connu de ce gabarit est **styles.content.get**, souvent utilisé en pratique, qui sélectionne les éléments de contenu à afficher entrés dans le backend via la colonne **Normal** :

```
styles.content.get = CONTENT
styles.content.get {
  table = tt_content
  select.orderBy = sorting
  select.where = colPos=0
  select.languageField = sys_language_uid
}
```

Vous n'êtes jamais obligé d'activer séparément le gabarit : il est intégré automatiquement si vous utilisez le gabarit **content (default)**. De nombreux paramètres sont configurés par des constantes dans ce gabarit (module : **Gabarit** → **Constant Editor** → **Content**).

5.5.3 cSet.*

cSet (default)

Cette option a été introduite pour améliorer l'efficacité lors de la création de sites Web ; elle continue d'être souvent utilisée en pratique, même si elle a été remplacée en grande partie par **cSet stylesheet**, et si des concepts récents tels que l'extension CSS **styled content** la rendent complètement superflue. Le gabarit contient seulement des définitions de constantes. L'idée centrale du gabarit est de rassembler les définitions de **styles.content.default** et de rendre disponibles de nouvelles constantes globales. Cela permet, entre autres, de définir en une entrée la fonte pour tous les objets de contenu. Les définitions rassemblées de cette façon sont facilement modifiables dans le Constant Editor, dans la catégorie **CSET**.

cSet stylesheet

Il s'agit d'une alternative moderne : ici, l'accent est placé sur l'utilisation de feuilles de style en cascade (CSS) insérées dans un fichier séparé (**media/scripts/defaultstylesheet.css**). Vous

remplacez les valeurs par défaut en en insérant d'autres dans un fichier que vous créez et dont vous spécifiez le nom dans le champ correspondant du Constant Editor.

```
# Exemple d'entrée dans la section Constants
content.stylesheet.file = fileadmin/styles/defaultstylesheet-modifie.css
```

Ce gabarit ne contient pas que des constantes, comme c'est le cas de **cSet (default)**, mais il change également plusieurs paramètres dans la configuration. En particulier, les balises `<font>` sont supprimées, les en-têtes de section remplacés par des en-têtes natifs HTML (h1-h6) et `<br/>` remplacé par `<p> </p>`.

5.5.4 frameset;*

```
frameset; top-page-right
frameset; top-page
frameset; top-left-page
frameset; top / left-page
frameset; page-bottom
frameset; left-page
frameset (+); top / left-adr-page
```

Ces modèles pour les sites Web basés sur des cadres (frame) sont utilisés en partie dans les gabarits des sites décrits ci-après. Le nom fait référence à la mise en page à réaliser ; par exemple, **frameset ; top-page-right** signifie : un cadre au-dessus, un cadre pour le contenu au centre (page) et un cadre à droite. La taille des cadres est fixée par des constantes. Certains gabarits de cadres peuvent s'emboîter. Le gabarit avec (+) ne génère pas une page entière mais sert d'extension à un autre gabarit de type **frameset**. Ces gabarits ont une valeur didactique car ils facilitent la compréhension de la manipulation de cadres en TypoScript. En pratique toutefois, la mise en page utilisée est rarement celle couverte par ces modèles.

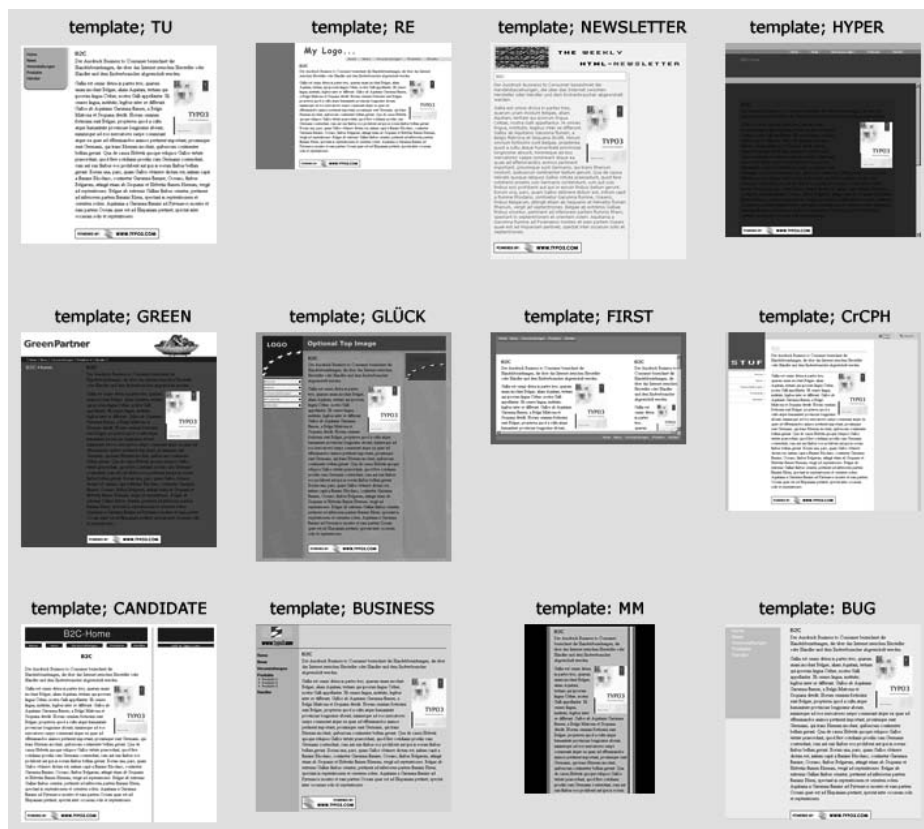
5.5.5 template;*

Les gabarits contenant ce préfixe sont des gabarits prêts à l'emploi, facultatifs pour l'affichage du site Web. Ils sont basés sur le gabarit standard **content (default)** pour la restitution du contenu.

Vérifiez que le gabarit standard désiré est au début de la liste des gabarits standards inclus. Vous pouvez faire quelques ajustements très facilement avec le Constant Editor.

Ces gabarits sont idéaux non seulement pour les exemples mais aussi pour les sites Web qui doivent être mis en ligne rapidement. Les exemples sont relativement anciens et sont basés soit sur des gabarits TypoScript purs, soit sur l'intégration de gabarits HTML ou encore sur des cadres. Un groupe de projet s'est formé au sein de la communauté TYPO3, avec pour but de créer de nouveaux gabarits munis d'une charte graphique moderne et répondant aux normes d'accessibilité. Ces gabarits seront alors déplacés dans une extension.

Figure 5.50:
Site Web basé sur
certains gabarits
standards



Les gabarits prêts à l'emploi actuels et leur application :

TU

TU est un gabarit sans cadre. Vous pouvez préciser une image d'en-tête et la couleur d'arrière-plan de la page. Le menu graphique sur le côté gauche comprend deux niveaux ; la couleur est modifiée lorsque la souris est positionnée sur un élément du menu ou pour les éléments de menu actifs. Vous pouvez ajouter des images en arrière-plan des éléments du menu et définir leur police de caractère TrueType, leur couleur et la taille de texte ainsi que la marge. Il y a aussi une option pour insérer des graphiques avant et après le menu (les fameux clear.gif).

RE

Ce gabarit se base sur des cadres. Le cadre supérieur contient un menu graphique pour le premier niveau, le cadre de gauche un menu graphique pour le second niveau. Le troisième cadre contient les éléments de contenu de la page. Il existe trois états pour les éléments de menu (images et couleurs en arrière-plan, couleur du texte, etc.), à savoir Normal, Rollover et Active. Si vous le désirez, vous pouvez définir une image en arrière-plan de chaque cadre et ajouter un graphique en dessous du menu de gauche.

NEWSLETTER

Ce gabarit sert aux pages sans menu. Comme son nom l'indique, il est prévu pour envoyer des bulletins d'information sous forme d'email HTML à l'aide de l'extension **Direct Mail module**. Le fichier gabarit HTML sur lequel il est basé détermine la mise en page. Vous pouvez choisir d'avoir une ou deux colonnes. Le logo dans l'en-tête n'est pas défini dans ce gabarit HTML mais il est configuré directement via TypoScript.

HYPER

HYPER est un gabarit basé sur des cadres composés d'un en-tête avec un menu de couches DHTML et sur un cadre à une colonne pour les éléments de contenu. Le titre de la page est affiché dans une image et vous pouvez spécifier un logo pour la section d'en-tête. L'apparence des pages est paramétrée par les feuilles de style.

GREEN

Ce gabarit d'une page se base sur un gabarit HTML. Cela signifie que tous les éléments tels que le menu principal, les sous-menus, l'image d'en-tête, le logo, le titre de la page et son contenu sont placés dans ce fichier avec les marqueurs correspondants. Vous pouvez modifier le gabarit HTML comme vous le voulez. Le menu est un menu popup DHTML dans lequel le premier niveau est de type graphique et le second de type texte. En option, un autre menu de texte peut être défini en troisième niveau. Vous pouvez également mettre le titre de la page en deçà du menu. Les images en arrière-plan des tableaux et de leurs cellules sont définies avec TypoScript.

GLÜCK

Ce gabarit standard est basé sur une table à trois colonnes. Le premier niveau du menu, de type texte, est dans la colonne de gauche, le second niveau dans la colonne de droite. Chaque colonne est optionnelle et peut être désactivée en utilisant TypoScript. Vous avez la possibilité de définir un logo, une image en arrière-plan et des éléments de contenu pour chaque colonne. Vous pouvez ajuster en détail les marges, les largeurs et les alignements. L'arrière-plan de la page et sa couleur sont aussi modifiables. Ce gabarit est entièrement configuré par TypoScript.

FIRST

FIRST est un gabarit basé sur des cadres dans lequel les cadres **Menu**, **Bottom** et **Page frame** sont emboîtés dans un autre cadre qui centre la page dans le navigateur. Les menus sont disposés graphiquement sur deux niveaux. Dans le cadre **Bottom** se trouve un menu général d'un répertoire. Déterminez la couleur d'arrière-plan et les dimensions des trois cadres ou intégrez éventuellement une nouvelle colonne avec des actualités associées à une page spécifique.

CrCPH

CrCPH est un gabarit d'une page basé sur un gabarit HTML. TypoScript est moins utilisé ici. Vous pouvez modifier de grandes parties de la mise en page en éditant le fichier gabarit avec votre éditeur HTML préféré. Les deux menus textuels sont aussi définis directement dans le gabarit HTML.

CANDIDATE

CANDIDATE est un gabarit d'une page contenant deux colonnes de contenu et un menu textuel à un niveau, situé en dessous de l'image d'en-tête dans la colonne principale. Le titre de la page (ou le sous-titre s'il en existe un) apparaît sur l'image d'en-tête, mais

vous pouvez également le désactiver. Vous pouvez aussi désactiver la colonne de droite ou l'afficher plutôt à gauche. Il vous est aussi possible de placer l'image d'en-tête sur les deux colonnes ou de changer cette image dans une sous-partie du site Web. La première image est affichée pour toute la section du site correspondant aux pages de premier niveau. La largeur et la couleur de l'espace entre les deux colonnes sont spécifiées ou réduites à zéro.

BUSINESS

BUSINESS est un gabarit basé sur des cadres avec une simple mise en page en deux colonnes. Le menu se trouve dans le cadre de gauche et le contenu de la page dans celui de droite. Vous pouvez définir les images en arrière-plan pour chaque cadre et choisir la largeur pour le cadre de gauche. Le menu est textuel et contient deux niveaux. Les balises de fontes sont ajustables dans les deux niveaux. Au deuxième niveau, vous avez la possibilité de placer des graphiques devant chaque élément de menu, qui peuvent changer lorsque la souris est placée à leur niveau. Dans l'en-tête de la page, le titre des pages à un niveau supérieur est indiqué en reprenant deux niveaux supérieurs à la page courante ; vous pouvez définir, comme pour les menus, les balises de fontes pour le titre de cette page. Il est possible de tracer une ligne en dessous du titre de la page et d'en définir la couleur.

MM

MM est un gabarit d'une page contenant dans la zone d'en-tête un menu basé sur des images. Vous pouvez définir des images en arrière-plan et la couleur de celui-ci, ainsi que la couleur et la largeur des bords. De plus, vous pouvez spécifier les éléments de contenu pour une seconde colonne éventuelle et déterminer la couleur et les images de son arrière-plan, de même que d'autres en-têtes standards et propriétés du texte. Vous pouvez ajouter un décalage entre les deux colonnes et choisir librement les images et la couleur de l'arrière-plan. Pour le menu, les options de configuration suivantes existent : couleur de l'arrière-plan, fonte TrueType, couleur de texte, taille et espacement de l'image.

BUG

BUG est un gabarit composé de cadres. Trois cadres sont disponibles : un au-dessus de la page, un à gauche et un à droite. Vous pouvez configurer la taille et la couleur de l'arrière-plan pour tous les cadres. Vous pouvez inclure un graphique dans le cadre du dessus et préciser l'apparence du menu graphique dans le cadre de gauche : image en arrière-plan pour l'ensemble du menu et pour les éléments de menus, effets de survol (pour la couleur du texte et/ou des graphiques avant ou après le titre de la page), polices de caractère TrueType, couleurs de texte, tailles, ombrages et espacements.

5.5.6 plugin.*

`plugin.tt_rating` [DEPRECATED]

...

`plugin.tt_board_list` [DEPRECATED]

`plugin.tipafriend` [DEPRECATED]

`plugin.postit1`

plugin.meta [DEPRECATED]
 plugin.feadmin.fe_users [DEPRECATED]
 plugin.feadmin.dmailsubscription [DEPRECATED]
 plugin.alt.xmlnewsfeed (89)
 plugin.alt.xml (96)
 plugin.alt.wap (97)
 plugin.alt.print (98)
 plugin.alt.plaintext (99)
 plugin.alt.pda (95)

Les plugins sont responsables de l'affichage et des caractéristiques des éléments de contenu spécifiques dans le frontend. Ils contiennent normalement leurs propres fonctions PHP. Les gabarits standards avec le préfixe `plugin.` se divisent en deux groupes. Le premier groupe reprend des éléments de contenu qui seront insérés dans le frontend lors du processus de restitution de contenu, par exemple `plugin.tt_news` ou `plugin.feadmin.fe_users`. Le second groupe (`plugin.alt.*`) génère des pages complètes qui sont traitées et affichées différemment des pages de contenu normales. Selon qu'il s'agit d'une version imprimée, d'une version texte, d'une version XML ou d'une version WAP pour les téléphones mobiles, le contenu est affiché différemment. Alors que la version pour l'impression est implémentée uniquement avec TypoScript, c'est un tout autre moteur de restitution qui est utilisé pour le plugin XML. Grâce à leur capacité à afficher du contenu dans plusieurs formats différents, la fonctionnalité des gabarits `plugin.alt.` évolue vers la publication « cross-media ».

Presque tous les plugins du premier groupe reçoivent l'étiquette `DEPRECATED` car lors de l'introduction de l'Extension Manager, tous les gabarits ont été déplacés vers les extensions correspondantes. Ces gabarits ne devraient pas être utilisés pour de nouveaux projets. Mais ils sont toujours dans la liste des gabarits standards, pour des raisons de rétrocompatibilité.

5.5.7 temp.*

`temp.tt_board (shared)` [DEPRECATED]

Les gabarits `temp` sont des gabarits d'aide qui servent de composants ou d'« extraits de code » à d'autres gabarits. `temp.tt_board (shared)` est contenu dans la fonction du plugin forum et sert de base à deux autres gabarits : `plugin.tt_board_list` et `plugin.tt_board_tree`. Il se charge de l'affichage général et de la configuration du forum TYPO3. Les deux autres gabarits du forum qui affichent des listes et des arborescences utilisent les paramètres de base de ce gabarit d'aide. Les trois gabarits sont déclassés car ils ont été déplacés dans l'extension Message board, twin mode (`tt_board`).

5.5.8 content.tt_*

`content.tt_address` [DEPRECATED]

Cette méthode est utilisée pour afficher directement les enregistrements de tables, ici `tt_address`. L'affichage des adresses ne requiert pas de plugin avec fonctionnalité PHP, mais est mis en forme par de simples objets TypoScript. Ce plugin est aussi déclassé car il a été remplacé par le gabarit de l'extension Address list (`tt_address`).

5.5.9 (example)

records (example)

De même que le gabarit `content.tt_address`, ce gabarit d'exemple montre comment le résultat du contenu des tables étendues (`tt_address`, `tt_links`, `tt_news`, etc.) peut être restitué simplement avec TypoScript. Vous ne devriez pas l'utiliser dans vos projets car il sert d'exemple.

5.5.10 language.*

`language.no` (norwegian) [DEPRECATED]

`language.nl` (dutch) [DEPRECATED]

`language.fr` (french) [DEPRECATED]

`language.dk` (danish) [DEPRECATED]

`language.de` (german) [DEPRECATED]

Les gabarits de langue servent à configurer différentes langues et constituent un ancien élément caractéristique de TYPO3. Ils sont presque devenus inutiles et ce, pas uniquement parce qu'ils ont été déplacés dans une extension. Même dans ces extensions, ils ne sont plus mis à jour, car le contrôle des langues est à présent enregistré dans des fichiers précis de certaines extensions. Dans les anciennes versions de TYPO3, ils servaient à la traduction des plugins et de certains objets de contenu comme les formulaires d'identification ou de recherche ; ils sont toujours présents pour la rétrocompatibilité.

5.6 Les bases de la mise en page – Concepts de gabarit

Lorsque vous développez une application Web avec TYPO3, vous êtes confronté au choix de la méthode à utiliser. Selon vos connaissances dans le développement de sites Web et les exigences du projet ou du client, vous allez décider si la mise en page doit être basée sur des tables HTML ou des feuilles de style en cascade (CSS), si vous utilisez des cadres, ... Vous devriez prendre ces décisions importantes indépendamment de TYPO3, puisque toutes les options sont réalisables avec ce CMS.

À l'étape suivante, vous décidez de la méthode à utiliser pour répondre aux exigences de la mise en page et pour insérer les éléments de contenu par TYPO3. Le fait qu'il y ait plusieurs méthodes empêche quelque peu de garder à l'esprit une vue d'ensemble et de faire le bon choix lors de votre première utilisation de TYPO3. Les méthodes de gabarit sont brièvement caractérisées ci-dessous ; ensuite, nous présenterons chaque mise en pratique.

5.6.1 Gabarits standards (gabarits statiques)

TYP03 vous fournit des gabarits complets prêts à l'emploi pour les sites Web, dénommés « gabarits standards ». Leur apparence et leurs fonctionnalités sont déjà configurées mais peuvent être ajustées dans une certaine mesure par des constantes. Ces gabarits sont considérés avant tout comme de très bonnes références pour apprendre à travailler avec TypoScript, mais ils peuvent bien entendu être aussi utilisés en production. Si vous devez respecter la charte graphique définie au sein de l'entreprise ou répondre à des exigences précises, le développeur atteindra rapidement les limites de la paramétrisation des gabarits standards.

5.6.2 Gabarits TypeScript purs

La mise en page de base d'un site Web est entièrement déterminée à l'aide de TypeScript, sans que vous deviez utiliser de fichiers HTML externes. Cette méthode vous permet de travailler de deux manières :

- Il est possible de spécifier rapidement une présentation sous forme de tableau en utilisant le cObject CTABLE qui divise un site Web en zones telles que topMenu, leftMenu, rightMenu, bottomMenu et content-cell.
- La mise en page est entièrement contrôlable via CSS en enveloppant tous les éléments de la page avec la balise HTML <div>. Cette technique est idéale pour implémenter des sites Web répondant aux normes d'accessibilité.

Les mises en page contrôlées par des tables ou des feuilles de style peuvent à présent être implémentées par l'intégration de gabarits HTML. Les gabarits TypeScript purs ont néanmoins certains avantages :

- Toute l'information qui contrôle l'affichage se trouve en un seul endroit et elle n'est pas stockée dans des fichiers HTML externes. Cela apporte plus de clarté et permet un contrôle complet.
- Vous pouvez construire le code HTML de manière modulaire en utilisant les champs **Constants** et **Setup** de TypeScript. Cela facilite les modifications réservées à certaines sections d'un site.
- Les gabarits TS peuvent être mis en cascade alors que les gabarits HTML ne le peuvent pas. Vous pouvez ainsi emboîter un gabarit de mise en page pour servir de base à d'autres gabarits, ou pour surcharger des propriétés de gabarits dans une section de l'arborescence.

L'utilisation d'un gabarit HTML externe est devenue à peine nécessaire (surtout pour les mises en page implémentées via CSS et qui n'utilisent pas de tables), c'est-à-dire qu'elle ne présente aucun avantage, puisque les zones de la disposition de base sont seulement marquées des balises <div> et que tout le reste est contrôlé via CSS.

5.6.3 Gabarits TypeScript et HTML

Même si les méthodes utilisant des gabarits TypeScript purs sont extrêmement flexibles et donc efficaces, les maquettes HTML sont de plus en plus utilisées de par leur simplicité. Pour cette raison, les développeurs ont l'opportunité d'intégrer des gabarits HTML externes via le cObject TEMPLATE. Le concept consiste à remplacer des zones du gabarit HTML, signalées par des marqueurs (subparts et marks), par des éléments de contenu de la base de données.

Les avantages de l'intégration de gabarits HTML sont les suivants :

- Du point de vue d'un développeur de sites Web, ces gabarits sont implémentés très rapidement avec un temps de préparation très court.
- Les mises en page de base se font avec des éditeurs externes.
- Pour un travail d'équipe, les graphistes et les développeurs TYPO3 peuvent travailler en parallèle si les zones de contenu et les fonctionnalités ont été précisées dès le départ.

5.6.4 Template Auto-Parser

Référence 591606

TYPO3 ne serait pas TYPO3 s'il n'essayait pas d'améliorer et de simplifier les procédures d'édition. C'est précisément l'objectif de l'extension **Template Auto-Parser**. Son utilisation permet l'analyse syntaxique des gabarits HTML à inclure pour marquer automatiquement les sous-parties (**subparts**) correspondantes. Elles sont identifiées par l'ID des éléments HTML tels que `<div>`, `span` et `td`. En outre, tous les chemins des feuilles de style et des images du gabarit HTML sont adaptés pour les faire pointer dans le bon dossier du système de fichiers de `fileadmin/`. De la documentation détaillée sur le Template Auto-Parser est fournie à la section 5.9.2 ainsi que dans le didacticiel « Modern Template Building, Part 1 » (voir la référence).

Les avantages et les désavantages du Gabarit Auto-Analyseur sont les suivants :

- La méthode s'adresse principalement à des agences et à des développeurs Web dans le cadre de l'intégration graphique d'une maquette HTML. Les résultats sont rapides car on ne doit pas ajouter à la main les marqueurs **subparts**. Mais les sentiments subjectifs du développeur TYPO3 sont que le Template Auto-Parser prend dans une certaine mesure tout le « contrôle » du processus.
- Si les rôles sont échangés lors du développement, il n'y a pas de risque que le graphiste supprime accidentellement les marqueurs **subparts** dans le gabarit HTML.

5.6.5 TemplaVoilà

Référence 003992

La méthode d'intégration de gabarit la plus récente est appelée **TemplaVoilà**. Il s'agit juste d'une extension (TemplaVoilà!) qui est apparue dans le contexte d'un projet assez important pour le groupe français Dassault Systèmes. Ce projet en est toujours pour l'instant à l'étape alpha et de nombreux développements doivent encore être accomplis. Mais cela vaut tout de même la peine de jeter un œil à ce concept et à ses perspectives, au moins pour le comparer à d'autres solutions.

TemplaVoilà permet d'implémenter le graphisme en quelques minutes (au lieu de quelques jours) pourvu qu'un gabarit HTML soit disponible. La zone de contenu de la page est divisée en zones pouvant contenir différents formats d'éléments de contenu. Cette technique remplace l'organisation de contenu en colonnes utilisée dans TYPO3. Dans ce but, une nouvelle option vient de s'ajouter à la liste déjà très longue des options pour la gestion des gabarits. Elle fournit une interface graphique au développeur pour la définition graphique des éléments de contenu en sélectionnant les fichiers de graphisme HTML.

Les avantages de TemplaVoilà sont les suivants :

- Cette approche rend la structure et la composition des blocs et des zones de contenu plus flexibles, avec tous les avantages d'un CMS et du contrôle extensif habituel du graphisme.
- De nouveaux éléments de contenu peuvent s'ajouter très rapidement à l'aide de gabarits et être utilisés immédiatement sans devoir les programmer.
- Vous pouvez définir des règles de contrôle pour déterminer les combinaisons possibles des éléments de contenu.

TemplaVoilà en est toujours au stade de développement. Des concepts techniques, qui étaient jusqu'à maintenant liés de près à l'organisation du contenu via des colonnes, doivent encore être implémentés.

Mais TemplaVoilà promet de présenter des avantages considérables dans le futur. Le concept est illustré plus en détail à la section 5.13.3.

Nous renvoyons tous ceux qui veulent essayer d'utiliser TemplaVoilà dans son état actuel au didacticiel très détaillé « Futuristic Template Building ».

5.7 Restitution du contenu

Indépendamment de la manière dont vous avez implémenté la mise en page de base, il y a différentes possibilités pour la restitution du contenu. Les méthodes de restitution dans le frontend ne sont pas restreintes à l'HTML.

Le contenu peut s'afficher comme du simple texte ou du XML. Mais même l'HTML a changé avec le temps de sorte que XHTML et accessibilité sont des mots-clés de nos jours.

TYPO3 offre plusieurs solutions et vous pouvez en développer d'autres vous-même. Évidemment, la mise en page de base doit correspondre au format du contenu et doit être omise lorsque besoin est (par exemple avec du texte pur).

Afin d'acquérir une compréhension de base de la restitution de contenu, nous devons revenir en arrière. Comme vous le savez déjà, TypoScript n'est pas un langage de programmation ; il sert plutôt à déterminer l'ordre du traitement des fonctions PHP à appeler et à déterminer leurs paramètres. Si vous définissez un objet de type `TEXT`, la fonction du même nom (dans ce cas), `TEXT()`, située dans le script PHP `tslib/class.tslib_content.php`, est appelée pendant le traitement du champ **Setup** de TypoScript, et le résultat correspondant est affiché. Les fonctions de restitution contenues dans le script ont été développées à un moment où l'utilisation de CSS était impensable, car soit les navigateurs ne supportaient pas du tout CSS, soit ceux-ci étaient inutilisables car l'implémentation comportait trop d'erreurs. Il y a donc eu une évolution certaine dans les concepts de restitution en TYPO3, évolution qui n'est pas encore terminée, et qui continuera certainement tant que la technologie du Web changera.

Les différentes variations dans la restitution HTML sont définies dans les gabarits standards suivants, qui vous sont déjà familiers :

`content (default)`

`cSet Stylesheet`

`css_styled_content`

`content (default)` est le plus ancien gabarit de restitution de contenu. Il utilise des balises `<font>` pour l'affichage. Le gabarit `cSet stylesheet` constitue une transition dans la restitution avec CSS en redéfinissant purement et simplement les valeurs de `content (default)` de sorte que plus aucune balise `<font>` n'est utilisée. Néanmoins, les fonctions de la classe `class.tslib_content.php` sont utilisées pour la restitution, en recourant fréquemment à des tables pour la disposition et à d'autres techniques qu'il serait plus judicieux d'implémenter avec CSS. L'extension `CSS styled content` a été développée pour cette raison. Elle utilise également les fonctions de restitution de `class.tslib_content.php`, mais remplace certaines fonctions par les siennes.

Le développement de `CSS styled content` n'est pas encore terminé, mais vous pouvez déjà utiliser cette extension pour vos propres projets.

En plus de l'affichage HTML, TYPO3 offre plusieurs autres formats prédéfinis utilisables pour les gabarits standards suivants :

```
plugin.alt.xmlnewsfeed (89)
plugin.alt.xml (96)
plugin.alt.wap (97)
plugin.alt.print (98)
plugin.alt.plaintext (99)
plugin.alt.pda
```

Les noms des gabarits donnent déjà quelques indications sur les formats qu'ils génèrent. Leurs propres scripts sont souvent utilisés pour l'affichage et vous n'utiliserez généralement plus les fonctions de la classe `class.tslib_content.php`.

5.8 Changer de gabarits avec `type/typeNum`

Il est souvent intéressant d'afficher du contenu dans le frontend sous différents formats ou différentes versions. Un exemple classique est le format pour l'impression. Mais il est aussi possible de fournir du contenu à d'autres sites Web avec du contenu local via XML.

Afin que différentes versions d'une page puissent être utilisées en même temps, TYPO3 fournit un système supportant simultanément plusieurs gabarits. Lors de la restitution, un seul gabarit est sélectionné.

Le gabarit TypoScript suivant devrait vous aider à comprendre ce concept :

```
temp.content = TEXT
temp.content.value = HELLO WORLD!
temp.content.wrap = <h1>|</h1>

page = PAGE
page.typeNum = 0
page.10 = COA
page.10 {
    10 = IMAGE
    10.file = media/uploads/typo3power1.gif
    20 < temp.content
}

plaintext = PAGE
plaintext.typeNum = 99
plaintext.config.disableAllHeaderCode = 1
plaintext.10 = COA
plaintext.10.stdWrap.stripHtml = 1
plaintext.10 {
    10 < temp.content
}
```

Les éléments de contenu à afficher (HELLO WORLD !) sont définis à l'aide de l'objet `temp.content` utilisé dans les deux configurations de page `page` et `plaintext` pour rendre les éléments de contenu disponibles. Comme nous le verrons plus tard, c'est `styles.content.get` qui est généralement utilisé pour récupérer les éléments de contenu dans la base de données.

Ainsi, avec `page` et `plaintext`, deux gabarits existent pour l'affichage de la page. `page` définit l'affichage HTML classique et contient aussi une mise en page de base, à savoir l'affichage du logo TYPO3 avant le contenu proprement dit.

La sortie HTML (abrégée) suivante est générée dans le frontend :

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>test plaintext</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <meta name="generator" content="TYPO3 3.8 CMS" />
</head>
<body bgcolor="#FFFFFF">

<h1>HELLO WORLD!</h1>
</body>
</html>
```

D'autre part, le gabarit `plaintext` devrait afficher du texte normal sans balises HTML. Il faut savoir que TYPO3 offre une meilleure option d'affichage de texte avec le gabarit standard `plugin.alt.plaintext` (99), et que ce gabarit est utilisé ici uniquement en guise d'exemple. Comme vous le voyez, la mise en page est différente (pas de logo). De plus, `config.disableAllHeaderCode=1` est configuré pour l'objet `PAGE`, ce qui a pour effet de bloquer l'affichage de l'en-tête HTML, et grâce à `stdWrap.stripHtml=1`, tout le code HTML est supprimé des éléments de contenu créés dans l'objet `COA`. Ce gabarit ne génère donc réellement que le texte **HELLO WORLD!**.

Mais comment peut-on sélectionner le gabarit `plaintext` ? La solution réside dans les valeurs `typeNum`.

```
page.typeNum = 0
```

```
plaintext.typeNum = 99
```

Si vous appelez la page dans le frontend et que vous ajoutez le paramètre `type=99` à l'URL (par exemple `http://www.example.org/?id=36&type=99`), TYPO3 choisit le gabarit *ad hoc*, c'est-à-dire celui pour lequel la valeur `99` est assignée à la propriété `typeNum`. Ce faisant, toutes les variations dans le résultat sont utilisables. Le même mécanisme s'applique aux cadres. Cette question est examinée plus en détail à la section 5.12.

Les nombres mentionnés entre parenthèses dans les gabarits standards `plugin.alt.*` sont les valeurs `typeNum` utilisées.

5.9 Création de gabarits TypeScript

Maintenant que nous avons établi les fondations de l'utilisation des gabarits et de TypeScript, nous pouvons exposer en pratique le développement des gabarits d'un site. Pour ce

faire, nous utiliserons le scénario BT3 qui a déjà été configuré au chapitre 3.10. Nous allons à présent donner une apparence au frontend en appliquant un scénario. Nous utiliserons à cet effet les différents sites à notre disposition dans l'exemple pour illustrer les possibilités du développement de gabarits. Un gabarit HTML a été fourni par le graphiste pour la mise en page de base. Il comprend déjà une structure de base et une ébauche des fonctionnalités à intégrer.

Figure 5.51:
Le gabarit graphique



La page contient plusieurs éléments :

1. Dans l'en-tête de l'application, un menu rootline montre le chemin de la page courante en partant de la page racine adéquate. Un lien permet d'obtenir la version imprimable.
2. L'en-tête contient le logo et des images qui changent en fonction du titre de la page et de l'endroit où on se trouve dans le site.
3. En dessous se trouve la navigation sous forme d'un menu horizontal.
4. Les éléments de navigation de niveau inférieur et ceux du menu méta sont situés dans une colonne sur la gauche.
5. La zone du contenu de la page est placée au milieu, avec une possibilité d'extension vers le bas de la page.
6. En option, des zones du site peuvent afficher des informations supplémentaires ou des actualités, dans une colonne sur la droite.
7. Pour terminer, il y a un pied de page avec, par exemple, un sigle copyright.

Référence 821706

Si vous voulez travailler sur les exemples, téléchargez les gabarits et les ressources nécessaires sous forme de documents TYPO3 (t3d) et d'une archive tar via la référence ci-contre et intégrez-les dans votre application.

Gardez TSref à portée de main, ou ouvrez l'assistant TS dans le backend (cf. section 5.4.2) : il vous permettra de naviguer commodément à travers les définitions d'objets.

Structure des gabarits

Pour être complet en abordant les exemples et vous fournir du matériel pour les reconstituer, les gabarits sont construits de manière très modulaire. Nous avons utilisé ici des gabarits en cascade. Plusieurs composantes des fonctions sont développées graduellement et intégrées à des gabarits parents. Cela vous permet de travailler efficacement et d'utiliser les gabarits plusieurs fois dans différents exemples.

Les gabarits en cascade sont obtenus de deux manières différentes :

- Traditionnellement, des gabarits sont intégrés à un autre gabarit à l'aide de **Include basis template**. Cette procédure a aussi été utilisée pour les gabarits standards fournis par TYPO3.
- Il est aussi possible d'emboîter des gabarits via des zones de page, c'est-à-dire de placer un gabarit sur une page, et donc de l'activer de ce fait automatiquement. Ce procédé est utilisé soit pour écraser des valeurs pour des parties précises du site Web, soit pour configurer une fonctionnalité spécifique pour ces parties.

On obtient une vue d'ensemble des gabarits répartis dans une application à partir de la page racine (cliquez sur le nom de votre installation situé à côté du globe et choisissez le module **Web** → **Gabarit**). Si vous avez reconstitué tous les exemples, votre résultat devrait ressembler à la figure suivante.

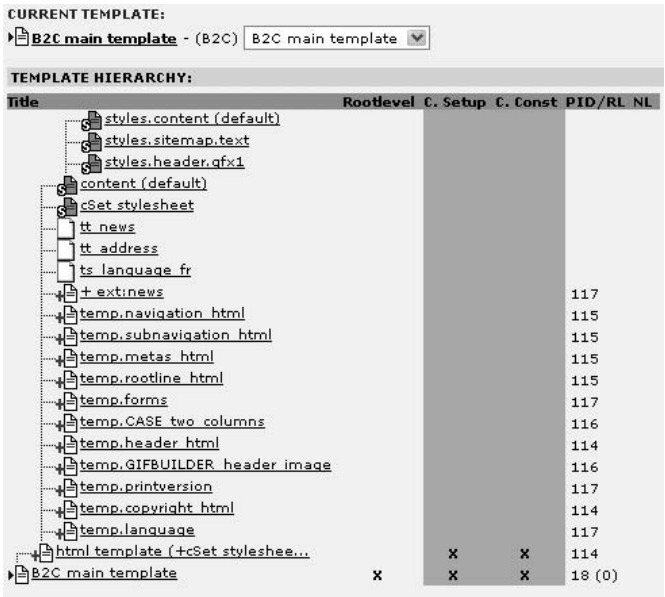
Template Tools			
This is an overview of the pages in the database containing one or more template records. Click a page title to go to the page.			
Page name	# Templates	Is Root?	Is Ext?
 Contenus	1	x	
 Portail	1	x	
 B2C-Accueil	4	x	x
 B2B-Accueil	1	x	
 Revendeurs	1		x
 Affiliation	1		x
 Evénements			
 Derniers événements	1		x
 B2E-Accueil	1	x	
 Fournitures Bureau	1		x
 Gabarits			
 Gabarits standards	12		x
 Premiers pas / Base	25	x	x
 Gabarits principaux	17		x
 Menus	14		x
 cObjects	3		x
 Fonctions	5		x
 Pas à pas	10		x

Figure 5.52:
Vue d'ensemble du
gabarit à partir de la
page racine

Nous avons défini le titre (**Website title**) uniquement dans chaque gabarit racine de site (« Portail », « B2C », « B2B », ...), c'est-à-dire dans les « gabarits principaux » (**B2C main template**, **B2B main template**, ...). La configuration TypeScript effective et l'insertion de gabarits standards sont réalisées dans les gabarits de base correspondants. Nous l'avons aussi fait au niveau des pages. Comme vous le voyez sur la figure précédente, les gabarits sont créés par exemple dans les pages « Revendeurs » ou « Affiliation » du site Web « B2B ». Ceux-ci permettent une configuration plus détaillée via **Include basis template**. Les gabarits de base utilisés dans ce chapitre sont tous situés dans le répertoire **Gabarits** du système et dans ses sous-répertoires. Cette procédure vous permet d'insérer, d'échanger et de tester rapidement les différentes méthodes à l'aide de ces enregistrements de gabarits.

La figure suivante nous montre un exemple de gabarits en cascade partant du site « B2C ».

Figure 5.53:
La hiérarchie des
gabarits dans le
Template Analyzer



5.9.1 TypeScript et gabarits HTML

Référence 905945

Nous voudrions illustrer l'utilisation de fichiers HTML pour la mise en page de base à partir du site « B2C ». L'intégration de gabarits HTML externes à un gabarit TypeScript est une méthode simple et souvent utilisée.

Voici les étapes que nous détaillerons :

- Comment préparer le document HTML ?
- Comment l'insérer à l'aide du cObject TEMPLATE ?
- Comment faire référence à certaines zones du gabarit ?
- Comment afficher le contenu dynamique dans le frontend ?
- Comment utiliser les bibliothèques de code ?

Le gabarit HTML

Le gabarit HTML est créé à l'aide de n'importe quel éditeur HTML ou éditeur de texte. Dans notre premier exemple, nous allons implémenter le site pour la zone « B2C » basée sur un gabarit HTML. Pour ce faire, nous utilisons une présentation avec des tables, sans recourir au cadre, ce qui correspond au gabarit graphique de la figure 5.51. La mise en page complète du site, comprenant les zones de fonction et de contenu, est alors conçue pour donner au développeur TypoScript des instructions claires pour l'étape d'implémentation.

À l'étape suivante, définissez les zones dont le contenu ou les fonctionnalités doivent être remplacées dynamiquement par TYPO3. Cela comprend les menus, les boutons, les zones de texte paramétrables et la zone de contenu.

Pour que TYPO3 puisse identifier ces zones, placez des balises spécifiques dans le gabarit HTML. Deux types de balises spécifiques sont prévues à cet effet : les sous-parties (subparts en anglais) et les marqueurs (markers en anglais).

Les sous-parties

- Les sous-parties (subparts) sont toujours utilisées par deux et renferment les sections du gabarit HTML qui sont complètement remplacées par le résultat de la configuration TypoScript.
- Le nom de la sous-partie est entouré par `###` et est sensible à la casse.
- Les sous-parties peuvent être entourées de commentaires HTML.
- Avant que les objets de contenu des sous-parties soient générés, toutes les sous-parties du tableau sont chargées dans le registre pour que vous puissiez y accéder si nécessaire.

Exemple :

```
<!--###CONTENT### start -->
...
<!--###CONTENT### stop -->
```

Les marqueurs

- Les marqueurs sont uniquement utilisés seuls (jamais par paires) et sont remplacés par le résultat de la configuration TypoScript.
- Le nom du marqueur est entouré par `###` et est sensible à la casse.
- Les marqueurs ne peuvent pas être à l'intérieur des commentaires HTML puisque ceux-ci ne sont pas supprimés.

Exemple :

```
###COPYRIGHT###
```


Préparation du gabarit HTML

Puisque les en-têtes HTML et les balises `<body>` sont généralement créés par TYPO3 depuis le frontend, seule la partie du gabarit HTML située entre les balises `<body>` doit être utilisée. Les balises des sous-parties sont insérées dans le code HTML dans ce but.

```
<body>
<!--###DOCUMENT_BODY### start -->
...
<!--###DOCUMENT_BODY### stop -->
</body>
```

Préparez le gabarit HTML de telle manière que tous les chemins d'image fassent référence aux répertoires du système de fichiers dans lesquels ils sont enregistrés, et que la référence soit relative au répertoire racine. Ils devraient aussi être accessibles à partir du module **Fichier** → **Fichiers** (par exemple `fileadmin/images/...`). Cela signifie que si vous placez le fichier HTML dans le répertoire racine du site Web et que vous y accédez, les images emboîtées seront visibles. Vous pouvez par conséquent accéder également à `clear.gif` qui se trouve dans le répertoire racine.

Figure 5.54:
Gabarit HTML appelé
via le répertoire
racine



Préparée de cette façon, l'esquisse HTML pour le gabarit apparaît à présent comme suit (en abrégé). Ici, les éléments de contenu à substituer ont été supprimés, pour rendre plus facile la lecture du document. À l'exception du marqueur `###COPYRIGHT###`, seules des sous-parties ont été utilisées.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Untitled</title>
  <link rel="stylesheet" type="text/css" href="fileadmin/styles/defaultstypesheet-angepasst.css">
</head>
<body>
<!--###DOCUMENT_BODY### start -->
<table width="800" border="0" cellspacing="0" cellpadding="0">
<tr>
  ...
</tr>
<tr>
  <td>&nbsp;</td>
  <td height="32" colspan="5" align="left" valign="middle">
    <table width="785" border="0" cellspacing="0" cellpadding="0">
      <tr>
```

```

        <td width="650" align="left" valign="middle">
            <!--###ROOTLINE### start -->Menu Rootline
            <!--###ROOTLINE### stop --></td>
        <td width="135" align="right" valign="middle">
            <!--###PRINTVERSION### start -->Version Impression
            <!--###PRINTVERSION### stop --></td>
        </tr>
    </table>
</td>
</tr>
<tr>
    ...
</tr>
<tr>
    <td width="15"></td>
    <td width="1" bgcolor="#CCCCCC" background="fileadmin/images/layout/1px_gray.gif"
    "></td>
    <td width="197" colspan="2" align="left" valign="middle"><
    /td>
    <td width="586" align="right">
        <!--###HEADER_IMAGE### start -->En-tête
        <!--###HEADER_IMAGE### stop --></td>
    <td bgcolor="#CCCCCC" background="fileadmin/images/layout/1px_gray.gif" width="1
    "></td>
</tr>
<tr>
    ...
</tr>
<tr>
    <td>&nbsp;</td>
    <td colspan="3" height="40"></td>
    <td colspan="2" height="40">
        <!--###NAVIGATION### start -->Navigation principale
        <!--###NAVIGATION### stop --></td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td bgcolor="#CCCCCC" background="fileadmin/images/layout/1px_gray.gif"><img src
    ="clear.gif" width="1" height="1" alt="" border="0"></td>
    <td align="left" valign="top">
        <!--###LEFT### start -->Navigation secondaire
        <!--###LEFT### stop --></td>
    <td>&nbsp;</td>
    <td align="left" valign="top">
        <!--###CONTENT### start -->Contenu
        <!--###CONTENT### stop --></td>
    <td width="1"></td>
</tr>
<tr>
    ...
</tr>

```

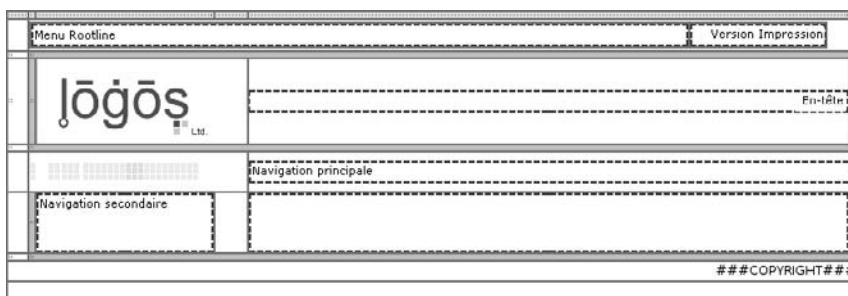
```

<tr>
  <td height="18" colspan="6" align="right">###COPYRIGHT###</td>
</tr>
<tr>
  <td height="10" colspan="6" align="right">&nbsp;</td>
</tr>
</table>
<!--###DOCUMENT_BODY### stop-->
</body>
</html>

```

À la figure suivante, vous voyez à nouveau le gabarit sous la forme d'un diagramme schématique. Ce gabarit sert à créer une mise en page typique à base de tables, avec des fichiers GIF transparents (en pointillés) en tant que balises spécifiques. Les zones avec des sous-parties sont marquées par des lignes brisées.

Figure 5.55:
Diagramme
schématique des
zones du gabarit
HTML



Dans cet exemple, de nombreuses sous-parties et marqueurs ont été insérés, de sorte que seule la mise en page de base demeure. Cela crée des zones de contenu qui peuvent toujours être structurées en utilisant TypoScript. L'avantage est que vous pouvez alors changer l'apparence des zones dynamiques à n'importe quel moment via les objets TypoScript, et que — dans nos exemples — nous pouvons recourir à différentes bibliothèques de code pour nos besoins d'illustration.

Le gabarit HTML doit désormais être enregistré dans le système de fichiers de l'application TYPO3, dans un répertoire sous `fileadmin/...`. S'il est placé dans le dossier racine, les références des images ne fonctionneront plus ; mais puisque le gabarit HTML est affiché avec `index.php` dans le frontend à partir du répertoire racine du site Web, les références fonctionneront.

Création de l'enregistrement de gabarit

Comme nous l'avons déjà mentionné, nous utilisons des gabarits en cascade dans les exemples, que nous insérerons à partir d'un gabarit racine.

Dans le répertoire du système **Gabarits/Gabarits principaux/**, créez un nouvel enregistrement de gabarit, **html template**, et insérez-le dans le gabarit racine de la page « B2C Accueil » (**B2C main template**) en tant que gabarit de base. **html template** représente la définition effective du gabarit de la page. Ce dernier comprend le gabarit content (**default**) et intègre aussi certaines bibliothèques de code sous forme de gabarits de base.

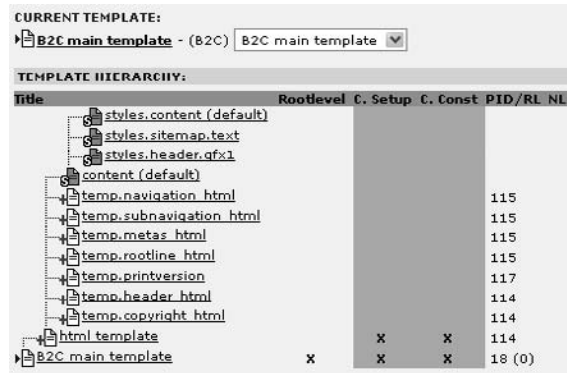


Figure 5.56:
Emboîtement futur
des gabarits TS dans
le Template Analyzer

Dans le champ **Setup** du gabarit TS `html template`, le TLO `page` est défini comme une page en l'assignant au type d'objet `PAGE` ; ce que nous avons déjà présenté à la section 5.2. Nous assignons la valeur `0` à la propriété `typeNum`. Puisque notre site n'est pas basé sur des cadres, cette assignation est suffisante.

```
page = PAGE
page.typeNum = 0
```

D'autres propriétés comme `bodyTagMargins` et `noLinkUnderline` spécifient que l'affichage de la balise `<body>` générée par TYPO3 se fait avec les paramètres `leftmargin="0"`, `topmargin="0"`, `marginwidth="0"`, `margin-height="0"`, et que les liens créés dans le contenu ne sont pas soulignés.

```
page.bodyTagMargins = 0
page.noLinkUnderline = 0
```

Si maintenant vous appelez une page à partir du site « B2C », vous verrez que le gabarit TS fonctionne. Alors que rien n'est encore affiché, les paramètres corrects ont été assignés à la balise `<body>` dans le texte source.

cObject TEMPLATE

À présent, pour insérer le gabarit HTML, utilisez le type d'objet `PAGE` qui reprend les objets de contenu dans une liste numérique (un tableau). La ligne `page.10=TEMPLATE` définit un objet de type `TEMPLATE` dans le chemin d'objet `page.10`. La propriété `template` que l'objet `page.10` a désormais acquise est alors définie comme un objet de type `FILE` et à travers la propriété `file`, cet objet fait référence au gabarit HTML dans le système de fichiers.

```
page.10 = TEMPLATE
page.10 {
    template = FILE
    template.file = fileadmin/templates/html_template.tpl
}
```

Avec ce code, le gabarit HTML est déjà affiché dans le frontend, mais les sous-parties et les marqueurs ne sont pas encore remplacés.

À ce stade, c'est une bonne idée de reconstruire la définition de l'objet **TEMPLATE** pour configurer les propriétés du type d'objet **TEMPLATE**, par exemple en ouvrant le champ TS **Setup** avec le module **Web** → **Gabarit** → **Info/Modify** et en utilisant l'assistant TypoScript présenté à la section 5.4.2. Il est clair que la propriété **template** peut prendre un cObject comme argument.

Remplacement des sous-parties et des marqueurs

Définissez d'abord la zone du gabarit HTML à utiliser pour l'affichage TYPO3. Étant donné que TYPO3 crée ses propres en-têtes et balises `<body>`, il s'agit de la sous-partie **###DOCUMENT_BODY###** située à l'intérieur de la balise `<body>` dans le gabarit HTML. Celle-ci délimite la zone du gabarit à interpréter pour le remplacement des sous-parties et des marqueurs à remplacer. Vous y faites référence avec la propriété **workOnSubpart** et le nom de la sous-partie **DOCUMENT_BODY** sélectionnée comme valeur. Les propriétés **subparts** et **marks** de l'objet **page.10** représentent des listes de cObjects avec les noms des sous-parties et marqueurs correspondants. Pour tester si les balises spécifiques données sont correctement référencées, elles sont définies ci-dessous en tant qu'objets de type **TEXT** et chacune d'elles reçoit un texte à afficher comme valeur.

```
page.10 = TEMPLATE
page.10 {
    template = FILE
    template.file = fileadmin/templates/html_template.tpl
    workOnSubpart = DOCUMENT_BODY
    subparts.HEADER_IMAGE = TEXT
    subparts.HEADER_IMAGE.value = zone pour l'image d'en-tête
    subparts.ROOTLINE = TEXT
    subparts.ROOTLINE.value = Menu rootline
    subparts.PRINTVERSION = TEXT
    subparts.PRINTVERSION.value = bouton d'impression

    subparts.CONTENT = TEXT
    subparts.CONTENT.value = zone de contenu

    subparts.NAVIGATION = TEXT
    subparts.NAVIGATION.value = zone de navigation principale
    subparts.LEFT = TEXT
    subparts.LEFT.value = zone de navigation secondaire et de metas (p. ex. informat
ion légale)
    marks.COPYRIGHT = TEXT
    marks.COPYRIGHT.value = bas de page avec copyright
}
```

Si une page du site « B2C » est appelée à ce moment dans le frontend, vous voyez que les sous-parties et les marqueurs ont été remplacés par le texte correspondant.

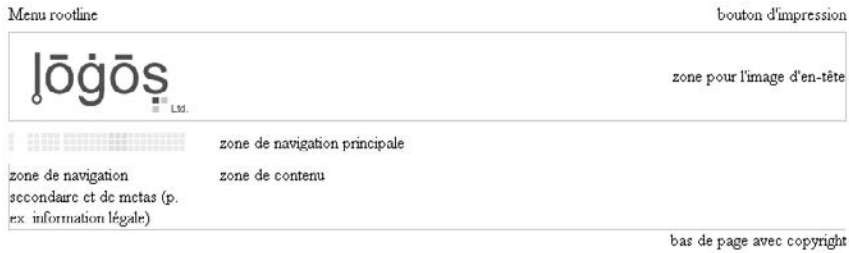


Figure 5.57:
Les marqueurs du
gabarit HTML ont été
remplacés

Insertion de contenu dynamique

À présent, le contenu des pages respectives devrait bien sûr être affiché. Puisque `content` (default) et `styles.content` (default) ont été insérés en tant que gabarits de base dans le gabarit TS html template, cela devient simple car dès cet instant, l'objet `styles.content.get` est disponible.

Extrait de `styles.content` (default) :

```
styles.content.get = CONTENT
styles.content.get {
  table = tt_content
  select.orderBy = sorting
  select.where = colPos=0
  select.languageField = sys_language_uid
}
```

Les éléments de contenu dans la colonne **Normal** (`colPos=0`) de la table `tt_content` sont sélectionnés et restitués avec le gabarit standard `content` (default) à l'aide de l'objet `styles.content.get`, car une configuration TS correspondante y est définie pour chaque type de contenu. La configuration est située sous `tt_content.*` et, puisque les noms sont identiques, elle est utilisée automatiquement par l'objet `CONTENT` pour restituer la table `tt_content`.

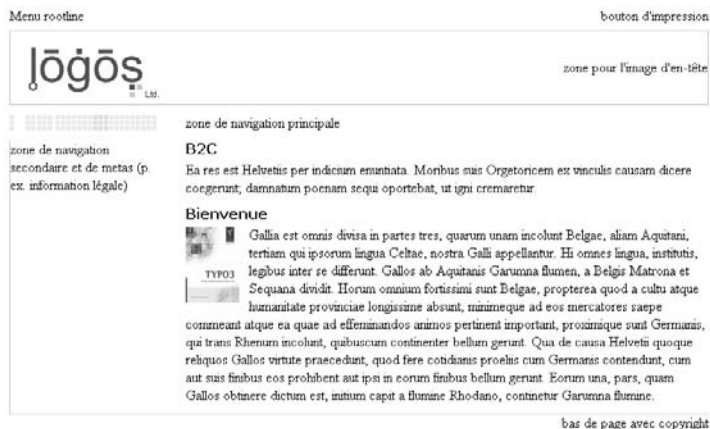
À présent, l'objet `styles.content.get` est aussi copié dans `page.10.subparts.CONTENT` : les éléments de contenu de la colonne **Normal** de la page courante sont restitués et insérés dans les sous-parties sous le nom `CONTENT`. Le contenu à afficher est mis dans un tableau via la propriété `wrap` du type d'objet `CONTENT`.

```
page.10 {
  ...
  subparts.CONTENT < styles.content.get
  subparts.CONTENT.wrap =<table width="586" cellpadding="0" border
="0"><tr><td>|</td></tr></table>
  ...
}
```

Notez que `CONTENT` dans le chemin d'objet `page.10.subparts.CONTENT` n'est que le nom de la balise de la sous-partie dans le gabarit HTML ; ceci ne signifie pas forcément que c'est un objet de type `CONTENT`. Mais en réalité, `page.10.subparts.CONTENT` est défini comme un objet `CONTENT` avec une copie de l'objet `styles.content.get`.

Les pages de l'application « B2C » restituent désormais le contenu de la page courante dans le frontend.

Figure 5.58:
Le contenu est affiché
dynamiquement



Pour mettre en forme l'affichage du contenu, les gabarits standards **cSet (default)** et **cSet Stylesheet** simplifient le marquage à l'aide du Constant Editor. Tous les deux définissent les styles dans `styles.content.default` à l'aide de constantes, ce qui facilite les changements à un niveau global. La différence entre les deux est que **cSet (default)** s'acquitte du marquage en utilisant la mise en forme HTML traditionnelle, alors que **cSet Stylesheet** utilise des définitions de feuilles de style en cascade qui sont enregistrées dans un fichier séparé. Selon la méthode que vous préférez, intégrez le gabarit standard approprié comme gabarit de base après `content (default)` dans le gabarit `TS html template`. Nous avons illustré ces deux possibilités tour à tour.

cSet (default)

Si **cSet (default)** est inséré, vous pouvez changer les valeurs par défaut avec le Constant Editor. Dans l'exemple, les valeurs standards de la catégorie **CSET** comme la couleur de l'arrière-plan, les polices de caractère et les tailles des différents éléments de contenu, ainsi que les en-têtes, sont tous remplacés.

```
#angepasste Konstanten des Standard-Templates cSet (default)
cSet.pageColor = white
cSet.tableCellColor = #E6B800
cSet.color = #666666
cSet.color1 = #0000CC
cSet.color2 = #E6B800

cSet.fontFace = Arial
cSet.fontFace.text = Arial

cSet.size1 = 1
cSet.size2 = 2
cSet.size3 = 3
```

Les changements sont sauves par le Constant Editor dans le champ Constant du gabarit sélectionné.

cSet Stylesheet

Ce gabarit standard se base sur la feuille de style `media/scripts/default-stylesheet.css` fournie dans l'installation de base de TYPO3. Copiez ce fichier et enregistrez-le dans un répertoire du système de fichiers sous `fileadmin/`. Spécifiez ensuite le nouveau chemin dans la catégorie **CONTENT** du Constant Editor.

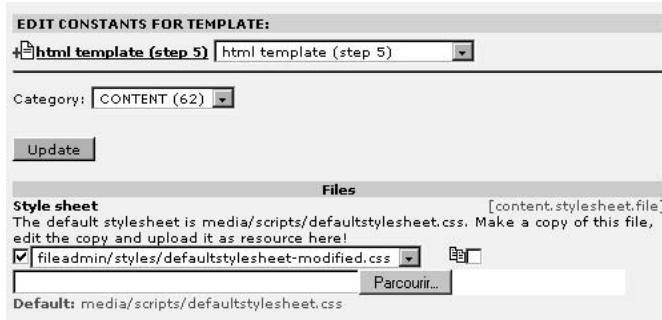


Figure 5.59:
Insertion du nouveau
fichier CSS

La nouvelle valeur de la constante est écrite dans le champ **Constants** du gabarit TS :

```
# constante de la feuille de style pour le gabarit standard cSet
content.stylesheet.file = fileadmin/styles/defaultstylesheet-modified.css
```

Vous pouvez à présent modifier les styles dans le fichier CSS ou en ajouter de nouveaux. Les modifications peuvent être effectuées soit avec un éditeur CSS spécial, soit directement avec TYPO3 (**Fichier** → **Fichiers**).

Étapes de construction avec des gabarits en cascade

Le gabarit TS insère le gabarit HTML et affiche les éléments de contenu mis à jour par les rédacteurs dans le backend. Toutefois, il contient encore des marqueurs spécifiques qui ne contiennent jusqu'à présent que du texte pour les tests. Ces balises seront remplacées par des bibliothèques de code et insérées en tant que gabarits de base. Nous discuterons des menus ultérieurement, dans le chapitre qui leur est consacré.

L'image de l'en-tête

Tout d'abord, il faut ajouter une image à la sous-partie **HEADER** qui, selon les zones de l'arborescence des pages, est spécifiée par les rédacteurs via le champ **Fichiers** des pages correspondantes. Pour ce faire, un gabarit TS particulier, `temp.header_html`, est créé. Il définit l'objet temporaire `header_image` du type d'objet **IMAGE**.

```
temp.header_image = IMAGE
```

Afin de spécifier l'endroit à partir duquel les images sont lues, utilisez la propriété `file.import` qui spécifie le répertoire où les images sont stockées via le champ **Fichiers**. La propriété `file` appartient au type de données `imgResource`. Cette propriété est à son tour une fonction qui contient la propriété `import`.


```
temp.header_image.file.import = uploads/media/
```

Nous avons maintenant précisé le dossier dans lequel les images sont situées, mais pas l'endroit où elles doivent être affichées. La propriété `import` appartient au type de données `path`, mais la fonction `stdWrap` est disponible, et donc la propriété `data` du type de données `getText` l'est aussi. Ceci permet d'afficher les valeurs des données enregistrées de manière interne par TYPO3 (tableau PHP), y compris les images assignées aux pages. Ainsi, `data = levelmedia:1`, `slide` affiche les données du type `levelmedia` du premier niveau dans l'arborescence des pages. Grâce au paramètre `slide`, TYPO3 parcourt l'arborescence des pages à partir de la page courante de niveau 1, jusqu'à ce qu'il trouve une image et l'affiche.

```
temp.header_image.file.import.data = levelmedia:1, slide
```

Nous regarderons de plus près la fonction `stdWrap`, mais nous avons vu ici combien elle est puissante rien qu'en affichant des données. Le point important ici est que si vous avez travaillé sur les exemples avec nous, vous aurez déjà compris quelque peu la manipulation de TSref (cf. figure 5.62).

Il ne vous reste plus qu'à insérer les images préparées dans les en-têtes de pages respectifs (type de page : **Advanced**) du premier niveau via le champ **Fichiers**.

Figure 5.60:
Fichier d'image inséré
dans l'en-tête de
page



Dans le gabarit TS `html template`, le chemin d'objet `temp.header_image` est assigné à la sous-partie `HEADER_IMAGE`.

```
subparts.HEADER_IMAGE < temp.header_image
```

De nombreuses images sont affichées, selon l'endroit de l'arborescence, dans l'en-tête de l'application du frontend via ces trois lignes de code TypoScript.

Figure 5.61:
Résultat dans le
frontend



Le texte dans le pied de page

Dans le bas de page de l'application, la note de copyright est insérée sous forme de texte qui peut être édité. Un autre gabarit TS est aussi créé ici (`temp.copyright_html`) et est inséré en tant que gabarit de base dans l'enregistrement de gabarit `html template`.

```
marks.COPYRIGHT < temp.copyright_html
```

IMAGE:

Property:	Data type:
file	imgResource

Datatype reference

Datatype:	Examples:	Comment:
resource	<i>From the resourcefield:</i> toplogo*.gif <i>Reference to filesystem:</i> fileadmin/picture.gif	1) A reference to a file from the resource-field in the template. You can write the exact filename or you can include an asterisk (*) as wildcard. It's recommended to include a "" before the fileextension (see example to the left). This will ensure that the file is still referenced correct even if the template is copied and the file will have it's name prepended with numbers!! 2) If the value contains a "/" it's expected to be a reference (absolute or relative) to a file on the file-system instead of the resource-field. No support for wildcards.
imgResource	Here "file" is a imgResource: file = toplogo*.gif file.width = 200 GIFBUILDER: file = GIFBUILDER file { ... (GIFBUILDER-properties here) }	1) A "resource" (see above) + imgResource-properties (see example to the left and object-reference below) Filetypes can be anything among the allowed types defined in the configuration variable \$TYPO3_CONF_VARS['GFX']['imagefile_ext'] (localconf.php). Standard is pdf,gif,jpg,jpeg,tif,bmp,ai,pcx,tga,png 2) GIFBUILDER-object

imgResource:

imgResource is properties that is used with the data type imgResource.

Property:	Data type:	Description:
import	path / stdWrap	value should be set to the path of the file with stdWrap you get the filename from the data-array Example: This returns the first image in the field "image" from the data-array: .import = uploads/pics/ .import.field = image .import.listNum = 0

stdWrap:

Property:	Data type:	Description:
Get data:		
data	getText	

Datatype reference

Datatype:	Examples:	Comment:
getText	get content from the \$cObj->data-array [header]: = field : header	This returns a value from somewhere in PHP-array, defined by the type. The syntax is "type : pointer"
	= leveluid : 0 Gets the value of the user defined field "user_myExtfield" in the root line (requires additional config in TYPO3_CONF_VARS to include field!)	leveltitle, leveluid, levelmedia: [levelTitle, uid or media in rootline, 0 - negative = from behind, " - slide" parameter forces a walk to the bottom of the rootline until there's a "true" value to return. Useful with levelmedia.]

Figure 5.62 :
Lecture du TRef

La partie de code `temp.copyright_html` génère un objet du même nom et du type d'objet `TEXT`. Sa propriété `value` reçoit comme valeur une constante et le résultat de l'objet est enveloppé d'une balise HTML `<font>`.

```
temp.copyright_html = TEXT
temp.copyright_html {
    value = {$copyright}
    wrap = <font face="Arial,Helvetica,sans-serif,sans-serif" size="1" color="#666666" class="copyright">|</font>
}
```

Il faut maintenant assigner une valeur à la constante dans le champ `Constants` du gabarit `TS` :

```
copyright = Copyright &copy; 2005
```

Ce texte est affiché dans le frontend à la fin de la page.

Figure 5.63:
Le gabarit `TS html`
template : seuls les
menus et la fonction
d'impression
manquent



Il n'est pas obligatoire d'utiliser la constante `copyright`, mais vous devriez pouvoir changer le sigle `copyright` en utilisant la valeur d'une constante.

L'application est encore étendue aux sections 5.10 et 5.11.

5.9.2 Le Template Auto-Parser

Référence 543497

L'utilisation du *Template Auto-Parser* est illustrée à l'aide de la zone « B2E » du site Web. Elle est similaire à l'intégration de gabarits HTML, sauf que l'insertion des sous-parties et des marqueurs ainsi que les chemins des images et des ressources sont pris en charge par l'extension *Template Auto-Parser* (`automaketemplate`). Celle-ci doit déjà être préalablement chargée et installée. Vous devriez aussi installer l'extension *CSS styled content* (`css_styled_content`) qui sera utilisée à la place du gabarit `standard content` (`default`).

Le gabarit HTML

Afin d'apporter quelques modifications à l'application, la mise en page du gabarit HTML dans la zone « B2E » n'est plus basée sur des tables, mais est marquée à l'aide de balises `<div>` dont l'apparence est entièrement définie par des feuilles de style en cascade (CSS)⁸. Ces quelques balises `<div>` peuvent aussi être créées avec TypeScript, comme nous le montrons à la section 5.9.3. L'Auto-Parser reconnaît les zones s'il trouve le paramètre `id` dans des éléments HTML tels que `<div>`, `<span>` et `<td>`. Le gabarit HTML a été préparé en conséquence et enregistré dans le système de fichiers. Si elles sont incluses dans le code HTML, les images et les autres ressources devraient être stockées dans des sous-répertoires relatifs correspondants. Les chemins vers les fichiers externes, comme les images, ne doivent pas être ajustés dans le gabarit HTML.

Référence 544119

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Untitled</title>
    <link href="fileadmin/styles/auto-parser-template.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="rootline">rootline</div>
<div id="header">
    <div id="logo">logo</div>
    <div id="headerimagetext">textimageentete</div>
</div>
<div id="navi">navi</div>
<div id="middle">
    <div id="subnavigation">sousnavi</div>
    <div id="content">contenu</div>
</div>
<div id="printversion">versionimpression</div>
<div id="footer"></div>
<div id="copyright">Copyright &copy; 2005</div>
</body>
</html>
```

Configuration du Template Auto-Parser

La hiérarchie du gabarit est identique à celle de la zone « B2C ». Un gabarit de base, `auto parser template`, est assigné au gabarit racine `B2E main template` qui est associé à la page racine du site « B2E ». Ce dernier gabarit contient le gabarit standard `CSS styled content` pour la restitution des éléments de contenu, via **Include static (from extensions)**.

Lorsque vous avez installé l'extension `Template Auto-Parser`, vous voyez apparaître un nouvel objet, `tx_automaketemplate_pi1` de type `USER` sous le `TLO` plugin.

⁸Bien sûr, l'efficacité de cette méthode ne devient apparente qu'avec des gabarits HTML plus complexes.

Figure 5.64:
L'extension Template
Auto-Parser dans
l'Object Browser



Référence 363442

Cela permet de faire appel au Template Auto-Parser et de le configurer. Les options possibles se trouvent dans la documentation de l'extension (voir la référence). Ajoutez le code suivant au champ **Setup** du gabarit TS auto parser template :

```
### Configuration du Template Auto-Parser : ###
plugin.tx_automakemtemplate_pi1 {
    content = FILE
    content.file = fileadmin/templates/auto-parser-template.tpl
    elements {
        BODY.all = 1
        BODY.all.subpartMarker = DOCUMENT_BODY
        HEAD.all = 1
        HEAD.all.subpartMarker = DOCUMENT_HEADER
        HEAD.rmTagSections = title
        DIV.all = 1
    }
}
```

La propriété `content` permet de choisir le gabarit HTML à utiliser dans le Template Auto-Parser. On donne ensuite l'instruction d'insérer les marques des sous-parties pour les balises `<body>`, `<head>` et `<div>`. Un `subpartMarker` (en majuscules) est configuré dans tous les éléments HTML de type `<body>` et `<head>`, de telle manière qu'ils soient reconnus par la suite par le cObject `TEMPLATE`. Puisque la section titre de la page est générée dynamiquement par TYPO3, la balise `<title>` est supprimée avec `elements.HEAD.rmTagSections=title`. S'il vous faut ajuster les chemins relatifs des images et des ressources du gabarit HTML à l'environnement TYPO3 à partir du répertoire racine, spécifiez un préfixe devant le dossier du système de fichiers. Nous ne l'avons pas fait dans l'exemple étant donné que seul le fichier CSS est impliqué, car la référence pointe déjà vers le bon répertoire.

```
#Ajoutez le préfixe suivant à tous les chemins relatifs :
plugin.tx_automakemtemplate_pi1.relPathPrefix = fileadmin/template/
```

Création de la page

Pour créer la page, le code TypeScript est construit de manière plus modulaire en créant des objets dans le chemin d'objet `temp.*`. L'objet `headerTemplate` du type d'objet `TEMPLATE` est créé pour intégrer les données de l'en-tête HTML. Sa propriété `template` reçoit comme référence l'objet `plugin.tx_automaketemplate_pi1`.

La propriété `workOnSubpart` spécifie que l'objet fait seulement référence à la zone du gabarit HTML qui a été entourée par la balise spécifique `DOCUMENT_HEADER`.

```
### cObject pour l'en-tête ###
temp.headerTemplate = TEMPLATE
temp.headerTemplate {
    template =< plugin.tx_automaketemplate_pi1
    workOnSubpart = DOCUMENT_HEADER
}
```

Dans l'exemple précédent, un fichier HTML contenant déjà des balises spécifiques a été lu par un objet du type `FILE` et est passé à l'objet `TEMPLATE` pour le traitement. La façon de procéder avec le Template Auto-Parser est identique, à ceci près que le plugin `plugin.tx_automaketemplate_pi1` configure les balises spécifiques automatiquement grâce à la configuration et qu'il ajuste également les références aux fichiers dans le processus. Vous pouvez donc assigner un id aux blocs du fichier HTML et y accéder dans la configuration TS, sans devoir insérer les balises spécifiques à la main.

L'objet `mainTemplate` est créé de la même manière, excepté qu'il fait référence à la zone du gabarit HTML entourée par la balise spécifique `DOCUMENT_BODY`. Des sous-parties sont à nouveau fournies avec le texte affiché en guise de test. On fait référence aux sous-parties grâce aux noms des id (observez le changement de casse) du gabarit HTML.

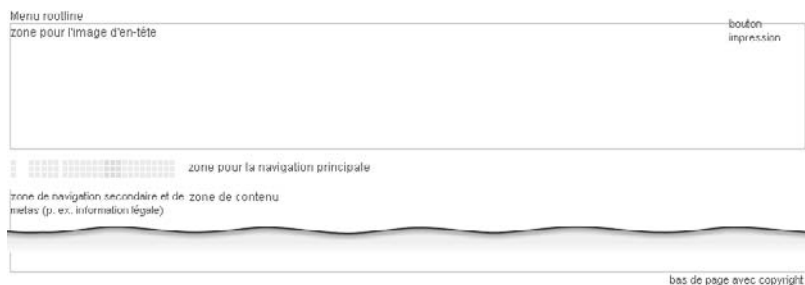
```
### cObject pour le corps de la page (body) ###
temp.mainTemplate = TEMPLATE
temp.mainTemplate {
    template =< plugin.tx_automaketemplate_pi1
    workOnSubpart = DOCUMENT_BODY
    subparts.rootline = TEXT
    subparts.rootline.value = Menu rootline
    subparts.printversion = TEXT
    subparts.printversion.value = bouton impression
    subparts.header = TEXT
    subparts.header.value = zone pour l'image d'en-tête
    subparts.navi = TEXT
    subparts.navi.value = zone pour la navigation principale
    subparts.subnavigation = TEXT
    subparts.subnavigation.value = zone de navigation secondaire et de metas (p. ex.
    information légale)
    subparts.content = TEXT
    subparts.content.value = zone de contenu
    subparts.copyright = TEXT
    subparts.copyright.value = bas de page avec copyright
}
```

Les pages créées dans la phase finale, ainsi que les objets temporaires, sont copiés dans les chemins d'objet `page.headerData.10` et `page.10`.

```
## Création de la page ###
page = PAGE
page.typeNum = 0
page.noLinkUnderline = 0
page.headerData.10 < temp.headerTemplate
page.10 < temp.mainTemplate
```

Lorsque vous affichez le frontend à ce stade, vous obtenez déjà la disposition de base contrôlée par CSS et les sous-parties correspondantes avec leurs éléments de contenu de test.

Figure 5.65:
L'affichage dans le
frontend à ce stade
de la construction



Si vous regardez le code source de l'affichage dans le frontend, vous voyez que le lien vers le fichier CSS situé dans l'en-tête reste intact, et qu'un certain nombre de balises spécifiques pour les sous-parties s'y trouvent toujours.

Figure 5.66:
Parties du résultat
dans le texte source

```
</head>
<body bgcolor="#FFFFFF">

<div id="rootline">Menu routline</div>
<div id="header">zone pour l'image d'en-tête</div>
<div id="navi">zone pour la navigation principale</div>
<div id="middle"><!--###middle### begin -->
  <div id="subnavigation">zone de navigation secondaire et de metas (p. ex. information légale)</div>
  <div id="content">zone de contenu</div>
<!--###middle### end --></div>
<div id="printversion">bouton impression</div>
<div id="footer"><!--###footer### begin --><!--###footer### end --></div>
<div id="copyright">bas de page avec copyright</div>

</body>
</html>
```

Si vous désirez que des marqueurs soient créés uniquement pour certains id, configurez le Template Auto-Parser en conséquence.

```
plugin.tx_automaketemplate_pi1.DIV.id {
    rootline = 1
    printimpression = 1
    ...
    copyright = 1
}
```

Étapes de construction avec des gabarits en cascade

Jusqu'à présent, nous n'avons affiché que du texte d'exemple avec ce gabarit. Ci-après, des objets TS seront développés pour des zones spécifiques, affichant le contenu approprié qui peut être incorporé au gabarit principal.

Affichage du contenu de la page

La colonne de contenu **Normal** a déjà été affichée avec le gabarit standard **content** (default) (via l'objet prédéfini **styles.content.get**). Le gabarit standard **CSS styled content** utilisé dans ce gabarit contient le même objet et peut être aussi utilisé ici.

```
temp.mainTemplate = TEMPLATE
temp.mainTemplate {
  ...
  subparts.content < styles.content.get
  ...
}
```

Création de l'en-tête

Contrairement à l'exemple de la section 58, le logo n'est pas contenu dans ce gabarit HTML. Nous devons le définir en utilisant TypeScript et l'image dans l'en-tête. C'est pourquoi un objet nommé **temp.header_tswrap_autoparser** du type d'objet **COA** a été créé dans le gabarit.

Le cObject **COA** (aussi appelé **COBJ_ARRAY**) peut contenir plusieurs objets dans une liste. Ainsi, l'objet **10** génère une image (type d'objet : **IMAGE**) localisée dans le système de fichiers via la propriété **file**.

```
temp.header_tswrap_autoparser = COA
temp.header_tswrap_autoparser {
  10 = IMAGE
  10.file = fileadmin/images/layout/logo.gif
```

Le logo devrait contenir un lien vers la page d'accueil du site Web. Pour ce faire, utilisez la fonction **stdWrap** que fournit le type d'objet **IMAGE** comme propriété.

Elle intègre la fonction **typolink** qui détermine (par la propriété **parameter**) à partir de quelles données le lien doit être créé. Le lien mène ici à la page dont l'ID est **75**.

```
10.stdWrap.typolink.parameter = 75
```

La propriété **parameter** est du type de données **string/stdWrap**. Une autre façon de procéder est de déterminer automatiquement l'ID de la page racine du site Web (level=0) grâce à la propriété **data** de la fonction **stdWrap**.

```
10.stdWrap.typolink.parameter.data = leveluid:0
```

L'image et le lien sont ensuite entourés d'une balise **<div>** par la propriété **wrap** de l'objet **stdWrap**, balise qui spécifie la position du logo par CSS.

```
10.stdWrap.wrap = <div id="logo">|</div>
```

20 est le second objet du **COA** créé par la copie de l'objet déjà existant **temp.header_image** dans ce chemin d'objet. **temp.header_image** provient du gabarit TS **temp.header_html** de l'exemple de la section 58 et est simplement réutilisé.

Pour ce faire, incluez ce gabarit dans l'enregistrement `auto parser template` en tant que gabarit de base. En outre, vous devez sélectionner les images désirées dans les pages correspondantes du premier niveau via le champ `Fichiers`.

Puisque l'application se base sur CSS, l'objet est enveloppé d'une balise `<div>`.

```
# image simple sans GIFBUILDER
20 < temp.header_image
20.wrap = <div id="headerimage">|</div>
}
```

Le texte dans le pied de page

Afin d'afficher le copyright au bas de la page en utilisant le gabarit TS `temp.copyright_html`, incluez-le en tant que gabarit de base et copiez l'objet dans `temp.mainTemplate.subparts.copyright`. La seule différence est que le texte ne doit pas être enveloppé par une balise `<font>`. Pour cette raison, la propriété `wrap` est supprimée à l'aide de l'opérateur `>`.

```
temp.mainTemplate = TEMPLATE
temp.mainTemplate {
    ...
    subparts.copyright < temp.copyright_html
    subparts.copyright.wrap >
    ...
}
```

Le gabarit a maintenant atteint le même niveau de développement que dans l'exemple précédent. La création des menus et des fonctions spéciales sera décrite plus tard. Avant cela, nous vous montrons à la section suivante comment construire des mises en page de base sans fichier HTML externe.

Référence 591605

Si vous appréciez de travailler avec le Template Auto-Parser, nous vous renvoyons une nouvelle fois au « Modern Template Building, Part 1 », un didacticiel très complet (voir la référence ci-contre).

5.9.3 Gabarits TypeScript purs

Dans la dernière méthode de création de gabarits TS que nous présentons ici, nous créons un gabarit de base entièrement avec TypeScript, sans avoir recours à des gabarits HTML.

Nous vous montrerons deux exemples utilisant la mise en page de l'application « B2B », l'un avec le concept d'enveloppe et avec des balises `<div>`, l'autre avec le cObject `CTABLE`. Les gabarits sont mis en cascade de la même manière que pour les applications « B2C » et « B2E ». Un gabarit de base, `ts wrap template` ou `ts CTABLE template`, est assigné au gabarit racine `B2B main template`. Définissez la page racine et intégrez les gabarits standards `content (default)` et `cSet stylesheet`, de même que d'autres composantes de code TS au fur et à mesure des explications.

Travailler avec wrap

Si nous nous référons à l'exemple du Template Auto-Parser, le gabarit HTML qui y est intégré peut être créé entièrement avec TypeScript. Si vous procédez ainsi, lors de l'analyse syntaxique,

aucun fichier HTML externe contenant des balises spécifiques n'est utilisé. Le remplacement des sous-parties et des marqueurs n'est pas nécessaire puisque tous les marquages HTML requis sont définis dans le gabarit TS.

Dans le gabarit TS `ts wrap template`, créez un TLO `page` du type d'objet `PAGE` et spécifiez l'apparence de la balise `<body>` ainsi que l'emplacement du fichier CSS dans le système de fichiers avec les propriétés `bodytag` et `stylesheets`.

Pour tous les éléments de contenu de la page, créez les éléments pertinents via une liste numérique, et entourez-les de balises `<div>` par la propriété `wrap`. L'apparence du site est aussi contrôlée par des feuilles de style en cascade (CSS).

```
page {
  10 = TEXT
  10.value = Menu rootline
  10.wrap = <div id="rootline">|</div>
  ..
}
```

Les composantes de code réutilisable déjà créées dans les gabarits TS `temp.header_tswrap_autoparser` et `temp.copyright_html` sont incluses dans l'enregistrement de gabarit avec le champ **Include basis template** et leurs propriétés sont passées aux objets respectifs de la liste numérique.

```
20 < temp.header_tswrap_autoparser
20.stdWrap.wrap = <div id="header">|</div>
```

Jusqu'ici, seul le contenu de la colonne **Normal** était affiché. Une seconde zone de contenu est à présent intégrée avec l'objet `60` qui affiche le contenu de la colonne de **Droite** via l'objet prédéfini `styles.content.getRight` du gabarit standard `styles.content` (default).

```
60 < styles.content.getRight
```

Maintenant, l'objet `60` est de type `CONTENT`. Afin d'entourer le contenu de balises `<div>`, pour respecter ce qui est défini dans la feuille de style CSS, nous utilisons la propriété `wrap` de la fonction `stdWrap`.

```
60.stdWrap.wrap = <div id="right"><div id="rightcontent">|</div>
<div id="rightfooter">&nbsp;</div></div>
```

L'enveloppe est activée avec la fonction `stdWrap`, car nous devons utiliser la propriété suivante :

```
60.stdWrap.required = 1
```

Si la propriété `required` est activée, `stdWrap` ne traite les autres propriétés activées que si une valeur est assignée à `stdWrap`. Le cas échéant, l'enveloppe ne s'affiche que si l'objet `page.60` de type `CONTENT` génère un résultat — c'est-à-dire s'il y a du contenu dans la colonne de droite. Dans l'affichage du frontend, une colonne de droite s'affiche via le CSS uniquement si le contenu existe également dans la colonne **Droite**.

La configuration de `ts wrap template` dans son étape actuelle de développement est la suivante :

```

page = PAGE
page.typeNum = 0
page.bodyTag = <body bgcolor="#FFFFFF" leftmargin="0" topmargin="0" marginwidth="0"
">
page.stylesheet = fileadmin/styles/ts-template-wrap.css

page {
# Top
## Menu rootline
10 = TEXT
10.value = Menu rootline
10.wrap = <div id="rootline">|</div>
## En-tête
20 < temp.header_tswrap_autoparser
20.wrap = <div id="header">|</div>
## Navigation principale
30 = TEXT
30.value = zone pour la navigation principale
30.wrap = <div id="navi">|</div>

# Gauche / menu secondaire
40 = TEXT
40.value = zone de navigation secondaire et zone de contenu metas (p. ex. inform
ation légale)
40.wrap = <div id="subnavigation">|</div>

# Contenu
50 < styles.content.get
50.stdWrap.wrap = <div id="content">|</div>

# Droite
60 < styles.content.getRight
60.stdWrap.wrap = <div id="right"><div id="rightcontent">|</div><div id="rightfo
oter">&nbsp;</div></div>
# la colonne de droite n'est affichée que si du contenu est défini
60.stdWrap.required = 1

# Version impression
65 = TEXT
65.value = bouton impression
65.wrap = <div id="printversion">|</div>

# Bas de page
## code pour permettre l'insertion en CSS d'un retour à la ligne
70 = TEXT
70.value = <div id="footer"></div>
## Copyright
80 < temp.copyright_html
80.wrap = <div id="copyright">|</div>
}

```

Travailler avec CTABLE

De simples présentations sous forme de tableaux sont rapidement créées avec TypeScript à l'aide du cObject **CTABLE**. Ce dernier crée un tableau standard, possédant jusqu'à cinq cellules. Une cellule pour les éléments de contenu (**c**) est éventuellement entourée de cellules pour la marge de gauche (**lm**), la marge de droite (**rm**) et les marges du haut (**tm**) et du bas (**bm**).

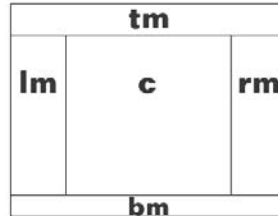


Figure 5.67:
Présentation en
tableau avec CTABLE

La mise en page du site de l'exemple n'est pas idéale pour la conversion avec **CTABLE** puisqu'elle contient plusieurs lignes et plusieurs cadres. Dans le gabarit de base **ts** **CTABLE** **template**, définissez ici aussi le TLO **page** avec les propriétés utilisées jusqu'à maintenant, à savoir **typeNum**, **bodyTagMargins** et **noLinkUnderline**.

```
page = PAGE
page.typeNum = 0
page.bodyTagMargins = 0
page.noLinkUnderline = 0
```

Dans cet exemple, nous avons divisé la mise en page en plusieurs objets, grâce à la propriété **numerical List** du type d'objet **PAGE**. L'objet 2 crée ensuite une table qui incorpore le menu rootline et le bouton pour la version d'impression. Le type d'objet **CTABLE** lui est assigné, ce qui rend toutes les propriétés des cObjects disponibles. Des paramètres supplémentaires sont activés pour la table avec les propriétés **tableParams** et avec **offset**, la distance au coin supérieur gauche est mise à 0.

```
##### rootline #####
page.2 = CTABLE
page.2 {
  tableParams = border="0" cellpadding="0" cellspacing="0" width="785"
  offset = 0,0
```

Les propriétés **c**, **lm**, **rm**, **tm** et **bm** sont du type de données **CARRAY +TDParams**, c'est-à-dire qu'elles représentent une liste numérique de cObjects. Vous pouvez faire passer vos propres paramètres dans les cellules de la table.

La cellule de contenu **c** pour le futur menu rootline est définie temporairement comme un objet de type **TEXT** ; une balise spécifique est définie avec sa propriété **value**.

```
c.10 = TEXT
c.10.value = Menu rootline
```

Le contenu de la cellule est centré :

```
c.TDParams = valign="middle"
```

Un cObject du type d'objet **CLEARGIF** servant à fixer la hauteur de la table à 32 pixels est introduit dans la cellule de gauche **lm**.

```
# Hauteur du menu rootline
lm.10 = CLEARGIF
lm.10.height = 32
```

Un texte est aussi configuré dans la colonne de droite **rm** pour le lien vers la version imprimable de la page.

```
rm.10 = TEXT
rm.10.value = bouton impression
rm.TDParams = valign="middle" align="right"
}
```

Les objets **page.10**, **page.20**, **page.30** et **page.40** sont également définis comme des tables. La cellule des éléments de contenu, **page.10.c**, affiche le logo et l'image dans l'en-tête de la page ; elle consiste en plusieurs objets se présentant sous la forme d'une liste numérique (COA).

```
##### en-tête #####
page.10 = CTABLE
page.10 {
  tableParams = border="0" cellpadding="0" cellspacing="0" width="785"
  offset = 15,0
  # Logo
  c.10 = COA
  #Table pour le le logo et l'image d'en-tête
  c.10.5 = HTML
  c.10.5.value = <table border="0" width="768" cellspacing="0" cellpadding="0"><tr
><td width="197">
  c.10.10 = IMAGE
  c.10.10.file = fileadmin/images/layout/logo.gif
  c.10.TDParams = align="left" valign="top" bgcolor="white"
  c.10.15 = HTML
  c.10.15.value = </td><td align="right">
  c.10.20 < temp.header_image
  c.10.30 = HTML
  c.10.30.value = </td></tr></table>
```

L'agencement des éléments est implémentée par les cellules **page.10.lm**, **page.10.tm**, **page.10.rm** et **page.10.bm** qui affichent les objets du type **IMAGE**.

```
# Agencement de l'en-tête
lm.10 = IMAGE
lm.10.file = fileadmin/images/layout/lpx_gray.gif
lm.TDParams = align="left" valign="top" bgcolor="#CCCCCC" height="125" width="1"

tm.10 = IMAGE
tm.10.file = fileadmin/images/layout/lpx_gray.gif
tm.TDParams = align="left" valign="top" bgcolor="#CCCCCC" height="1" width="785"
```

```

rm.10 = IMAGE
rm.10.file = fileadmin/images/layout/lpx_gray.gif
rm.TDParams = align="left" valign="top" bgcolor="#CCCCCC" height="125" width="1"

bm.10 = IMAGE
bm.10.file = fileadmin/images/layout/lpx_gray.gif
bm.TDParams = align="left" valign="top" bgcolor="#CCCCCC" height="1" width="785"
}

```

page.20 affiche la navigation principale.

```

##### Navigation principale #####
page.20 = CTABLE
page.20 {
    tableParams = border="0" cellpadding="0" cellspacing="0" width="785"
    offset = 15,0
    c.5 = HTML
    c.5.value = <table border="0" width="785" height="40" cellspacing="0" cellpaddin
g="0"><tr><td width="197">
        c.10 = IMAGE
        c.10.file = fileadmin/images/layout/balken_gelb.gif
        c.10.params = hspace="0" vspace="0"
        c.15 = HTML
        c.15.value = </td><td>
        c.20 = TEXT
        c.20.value = zone pour la navigation principale
        c.TDParams = align="left" valign="middle" bgcolor="#FFFFFF" height="40"
        c.30 = HTML
        c.30.value = </td></tr></table>
    }
}

```

La table page.30 met en place le menu secondaire et la zone de contenu par le biais de page.30.lm et page.30.c. Ceux-ci sont entourés par les objets HTML page.25 et page.35 qui affichent la ligne grise sur la page de gauche.

```

##### Sous-navigation et contenu #####
#Ligne de droite et de gauche sous forme de table
page.25 = HTML
page.25.value = <table border="0" cellspacing="0" cellpadding="0"><tr><td width="
15"></td><td width="
1" bgcolor="#CCCCCC" background="fileadmin/images/layout/lpx_gray.gif"></td><td>

page.30 = CTABLE
page.30 {
    tableParams = border="0" cellpadding="0" cellspacing="0" width="767"
    offset = 0,0
    c.10 = COA
    c.10.5 = HTML
    c.10.5.value = <table border="0" cellspacing="0" cellpadding="0" width="586"><tr>
    c.10.10 < styles.content.get
    c.10.10.wrap = <td align="left" valign="top">|<br /><br /></td>

```

```

c.10.25 = HTML
c.10.25.value = </tr></table>
# espace pour que la cellule soit aussi incluse si son contenu est vide
c.10.stdWrap.ifEmpty = &nbsp;
# décalage par rapport au copyright
c.20 = HTML
c.20.value = <br /><br />
c.TDParams = align="right" valign="top"

lm.10 = TEXT
lm.10.value = zone de navigation secondaire et de metas (p. ex. information légale)
# hauteur de 400px, même sans contenu
lm.TDParams = width="197" align="left" valign="top" height="400"
}

#Fin de la table
page.35 = HTML
page.35.value = </td></tr></table>

```

Le pied de la page avec le copyright est créée par l'objet table `page.40`.

```

##### Copyright #####
page.40 = CTABLE
page.40 {
    tableParams = border="0" cellpadding="0" cellspacing="0" width="785"
    offset = 15,0

    c.10 < temp.copyright_html
    c.TDParams = align="right" valign="middle" bgcolor="#FFFFFF" height="18"

    tm.10 = IMAGE
    tm.10.file = fileadmin/images/layout/lpx_gray.gif
    tm.TDParams = align="left" valign="top" bgcolor="#CCCCCC" height="1" width="785"

    bm.10 = CLEARGIF
    bm.10.height = 30
}

```

Comme vous le voyez, le code TypoScript est devenu assez long : `CTABLE` convient mieux à des mises en page plus simples.

À présent, toutes les méthodes de gabarit ont été passées en revue. Celle que vous choisirez pour concevoir votre application dépendra de vos préférences et de vos compétences. Dans la section suivante, nous introduisons les menus qui affichent automatiquement l'arborescence des pages pour la navigation à travers le site.

5.10 Menus

Les menus ne sont pas seulement à la base du concept de navigation dans votre site Web, ils constituent également des éléments de graphisme importants. Avec TYPO3, vous pouvez

créer différents types de menus grâce au TypoScript, pour afficher sous différentes formes la structure du site. Certaines fonctionnalités sont déjà incluses pour la création de menus basés sur du texte, sur des images, sur des compositions d'images, ou des menus de sélection.

Dans le frontend, les menus sont créés dynamiquement grâce à l'arborescence des pages. Si, par exemple, vous créez une nouvelle page dans le backend, elle est automatiquement reprise dans le menu correspondant une fois qu'elle est mise en ligne.

Dans les menus, les types de page affichés sont généralement **Standard**, **Avancé**, **URL externe**, **Raccourci** et **Point de montage**, pourvu que TypoScript n'impose aucune restriction. La balise spécifique **Délimiteur** doit être définie séparément pour être affichée. Les types de page **Hors menu**, **Dossier Système** et **Corbeille** ne sont pas inclus.

Niveaux

Puisque normalement, seul le plan du site affiche toutes les pages d'un site Web, vous devez définir dans la plupart des menus les pages de début et de fin du menu. Une manière de les définir réside dans les fameux *niveaux*. Le rootline du site Web, en tant que premier niveau, correspond au niveau 0. Le niveau suivant est le niveau 1, etc.

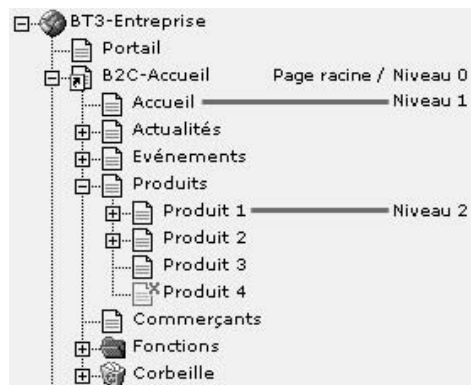


Figure 5.68:
Niveaux dans
l'arborescence des
pages

Le rootline

Un concept important, souvent utilisé dans le contexte des menus de TYPO3, est celui des *rootline*. Le terme désigne également le niveau le plus bas dans l'arborescence des pages. En sélectionnant le rootline (le globe), vous pouvez afficher par exemple les gabarits standards dans le module **Web** → **Liste**. Ce terme a toutefois un autre sens lorsqu'on parle de menus, ce qui peut amener une certaine confusion. Ici, le rootline est une liste de pages, obtenue en parcourant les pages depuis la page courante jusqu'à la page racine. À la figure 5.68, le rootline de la page « Produit 2 » serait : Produit 2, Produits, B2C Accueil.

Nous vous montrons ci-dessous comment configurer les différents types de menus avec TypoScript. Dans l'application exemple BT3, ils sont sauvegardés dans les enregistrements de gabarit dans le Dossier Système **Gabarits/Gabarits Principaux/Menus/**. Pour l'affichage dans

les sites Web « B2C », « B2B » et « B2E », les menus désirés sont inclus dans les gabarits principaux *ad hoc* en tant que gabarits de base. Les balises spécifiques dans les gabarits principaux sont remplacées par les objets correspondants.

Exemple : dans le site Web « B2C », le gabarit principal `html template` définit la page. Intégrez le menu que vous avez créé dans l'enregistrement de gabarit avec **Include basis template**.

Dans le chemin d'objet `page.10.subparts.NAVIGATION` qui jusqu'ici contenait un marqueur,

```
page.10 {
  ...
  subparts.NAVIGATION = TEXT
  subparts.NAVIGATION.value = zone de navigation principale
  ...
}
```

l'objet de menu est désormais copié. Remarquez que c'est bien l'objet du menu qui est copié et non l'enregistrement de gabarit, même si ces deux entités portent généralement le même nom dans les exemples.

```
...
subparts.NAVIGATION < temp.navigation_html
...
```

Si l'objet menu avait déjà été créé, il serait visible sur le site Web.

5.10.1 Le cObject HMENU – propriétés générales des menus

Référence 460074

Quel que soit le menu que vous vouliez publier sur votre site Web, il sera créé avec le cObject HMENU (*menu hiérarchique*).

Voici un exemple simple de création d'un menu pour les pages de niveau 0 dans lequel les entrées sont affichées les unes en dessous des autres, puisque chaque élément de menu contient une balise `<br />`.

```
temp.navigation = HMENU
temp.navigation.entryLevel = 0
temp.navigation {
  1 = TMENU
  1.NO.allWrap = |<br />
}
```

Le HMENU s'occupe de rassembler l'information de la page. La restitution effective des entrées du menu est réalisée par les sous-objets qui sont inclus dans une liste numérique. Par opposition avec les types d'objet PAGE et COA, la numérotation revêt ici une signification spéciale. Les propriétés 1, 2, 3, etc. représentent les niveaux des menus correspondants. La propriété `entryLevel` précise le niveau de l'arborescence des pages par rapport au rootline à partir duquel vous commencez à compter les niveaux des menus. La configuration TS ci-dessus va donc créer un menu à un niveau commençant au niveau 0.

Notez que dans notre exemple, nous utilisons un objet de menu TMENU qui crée des liens de type texte. Comme nous l'avons déjà mentionné, d'autres types de menus sont supportés par TYPO3.

```
temp.navigatation.1 = TMENU
# Le deuxième niveau du menu est un menu graphique.
temp.navigatation.2 = GMENU
```

Les types de menus suivants sont disponibles dans TYPO3 :

TMENU

Affiche un menu basé sur du texte.

GMENU

Un **GMENU** est un « menu graphique ». Les images sont générées automatiquement à partir des titres des pages.

GMENU_LAYERS/TMENU_LAYERS

Les **GMENU_LAYERS** et les **TMENU_LAYERS** ajoutent des propriétés supplémentaires à l'objet de menu **GMENU** ou **TMENU** correspondant, et l'affichent avec des couches DHTML. À cause des différentes interprétations des standards par les différentes versions des navigateurs, les objets de menu ne sont pas utilisés pour les anciens navigateurs et ne le sont que dans une certaine mesure pour les nouveaux.

GMENU_FOLDOUT

Un **GMENU_FOLDOUT** étend l'objet de menu **GMENU**. Il crée un menu à deux niveaux, s'étend ou se referme sans devoir recharger la page.

IMGMENU

L'objet **IMGMENU** crée un menu basé sur une image avec des zones sensibles.

JSMENU

Un **JSMENU** crée un menu basé sur une liste de sélection.

Outre la propriété **entryLevel**, un **HMENU** admet plusieurs autres propriétés qui contrôlent l'affichage du menu. Le nombre d'entrées du menu est spécifié via les propriétés **minItems** et **maxItems**, et avec **begin**, vous définissez la première entrée du menu. Avec **excludeUidList**, vous pouvez exclure certaines pages du menu.

special est une propriété puissante qui vous aide à définir les menus selon une structure différente de celle de l'arborescence des pages. Vous pouvez créer des menus qui affichent le rootline, montrent les pages contenant certains mots-clés ou donnent une liste des sous-pages des pages spécifiées. Toutes les options possibles sont passées en revue à la section 5.10.8.

Les objets de menu eux-mêmes (**TMENU**, **GMENU**, ...) ont des états communs mais ont aussi leurs propres états. Ils possèdent par exemple tous le statut **addParams**, qui ajoute des paramètres supplémentaires au lien du menu.

Référence 401761

À l'exception du **JSMENU**, tous les autres objets de menu de la série **TMENU**, **GMENU** et **IMGMENU** partagent des états comportant des éléments de menu. **NO** (normal) représente le cas normal qui doit toujours être défini comme état par défaut. Tous les autres états doivent être activés pour être utilisés, et leur comportement doit être spécifié individuellement.

Référence 478459

NO

L'élément du menu dans l'état normal **NO**. La définition est obligatoire.

IFSUB

Le statut **IFSUB** est actif si l'élément de menu a des sous-pages.

ACT

L'élément de menu a le statut **ACT** s'il est situé dans le rootline de la page affichée.

ACTIFSUB

Si l'élément de menu est situé dans le rootline de la page actuelle et s'il a des sous-pages à afficher dans un menu, alors il a le statut **ACTIFSUB**.

CUR

La page courante a le statut **CUR**.

USR

L'état **USR** s'applique aux pages dont l'accès est restreint par les groupes d'utilisateurs du frontend. Il a priorité sur les états **IFSUB**, **ACT** et **CUR**.

SPC

Le statut **SPC** s'applique en particulier au type de page **Spacer** et il est utilisé spécialement pour structurer les menus. **SPC** ne prévoit pas de fonctionnalité pour les effets de survol puisqu'il ne contient pas de lien.

USERDEF1

Les états **USERDEF1** et **USERDEF2** sont définis dans un script séparé (voir les propriétés **HMENU/special = userdefined** et **.itemArrayProcFunc**).

USERDEF2

Voir **USERDEF1**.

RO

Permet de créer une variation des effets de survol (rollover) pour l'élément de menu. Le statut **RO** n'est pas disponible pour beaucoup d'objets et doit donc être ajouté à la référence de ces objets.

Les éléments de menu sont introduits ci-dessous en se servant de l'application qui nous sert d'exemple. Seule une partie des propriétés disponibles seront utilisées. Consultez le document **TSref** pour obtenir une vue d'ensemble de toutes les autres propriétés.

5.10.2 Menus de texte (TMENU)

Référence 623534

Un **TMENU** crée un menu de texte à partir du titre de page ou de navigation de l'en-tête. Les titres sont automatiquement affichés sous forme de liens. C'est vous qui en configurez l'apparence. L'avantage principal des menus basés sur du texte est leur rapidité de chargement.

Nous voudrions mettre en évidence une caractéristique particulière de l'objet de menu **TMENU** : les propriétés **before** et **after** ont les mêmes propriétés que **stdWrap**. Mais la sous-propriété **field** n'est pas disponible directement pour ces dernières. Ceci est dû à une erreur de conception qui a été laissée dans **TYPO3**, par souci de rétrocompatibilité. Pour cette raison, vous devriez éviter cette propriété et utiliser à sa place la propriété **data** (par exemple **before.data=page:title**).


```

background: url(../images/icons/link.gif) no-repeat left;
text-decoration: none;
color: #666;
}
...

```

Le résultat dans le frontend est le même que celui de l'application « B2C ».

Exemple : sous-navigation dans « B2C »

Un TMENU est implémenté dans le gabarit `temp.navigation_html` pour la sous-navigation. Les éléments de menu des premier et deuxième niveaux sont arrangés en tableau et séparés par différentes lignes. L'état actif du premier niveau se distingue grâce au titre en majuscules. Pour le deuxième niveau, de petites icônes sont affichées devant les titres.

Dans le frontend, la navigation secondaire sur deux niveaux et les effets de survol apparaissent comme à la figure 5.70.



Figure 5.70:
La navigation
secondaire TMENU
sur deux niveaux

La navigation secondaire commence au niveau 1, ce qui est traduit par le code `entryLevel=1`.

```

temp.subnavigation_html = HMENU
temp.subnavigation_html.entryLevel = 1

```

Les pages méta « Accueil », « Impression », « Plan du site » et « Contact » du Dossier Système Fonctions ne sont pas affichées si vous donnez une liste d'ID de pages à la propriété `excludeUidList`. Toutefois, il est plus facile de donner à ces pages le type **Not in menu**.

```
temp.subnavigation_html.excludeUidList = 122,121,120,119
```

Le niveau 1 du menu est un TMENU.

```
temp.subnavigation_html.1 = TMENU
```

Ce niveau est mis en forme par une table à trois colonnes.

```
temp.subnavigation_html1.1.wrap = <table width="170" cellspacing="0" cellpadding="0"
" border="0"><tr><td width="7"></td><td width="11"></td><td width="152"></td></tr></table>
```

Les états **NO** et **ACT** sont configurés pour que chaque élément de menu soit affiché, via la propriété `allWrap`, dans une ligne différente de la table, et que l'image soit une ligne séparatrice. La propriété `wrapItemAndSub` permet d'inclure les éléments de sous-menu de niveau 2, mais ce n'est pas ce que nous voulons faire ici.

```
temp.subnavigation_html1.1 {
    NO {
        linkWrap = <font face="Arial,Helvetica,sans-serif,sans-serif" size="2" color="#666666" class="subnavi_no"> &nbsp;&nbsp; |</font>
stdWrap.case = lower
allWrap = <tr><td&nbsp;&nbsp;&nbsp;</td><td colspan="2"><br /> |</td></tr>
ATagBeforeWrap = 1
    }
ACT < .NO
ACT.stdWrap.case = upper
ACT = 1
}
```

Le second niveau du menu est aussi un **TMENU**. Les éléments de menu du second niveau ont leurs propres icônes et sont disposés par une mise en page HTML grâce à la propriété `allWrap`.

```
temp.subnavigation_html1.2 = TMENU
temp.subnavigation_html1.2 {
  NO {
    linkWrap = <font face="Arial,Helvetica,sans-serif,sans-serif" size="2" color="
#666666" class="subnavi2_no"> |</font>
    stdWrap.case = lower
    allWrap = <tr><td colspan="2">&nbsp;</td><td><br /> |</td></tr>
    beforeImg = fileadmin/images/icons/subnavi2_no.gif
    beforeROImg = fileadmin/images/icons/subnavi2_ro.gif
    RO = 1
    ATagBeforeWrap = 1
  }
  ACT < .NO
  ACT.beforeImg = fileadmin/images/icons/subnavi2_act.gif
  ACT = 1
}
```

Incorporez le gabarit TS dans le gabarit principal du site « B2C » et retournez à nouveau dans le frontend afin de visualiser le résultat .

5.10.3 Menus graphiques (GMENU)

Référence 193130

Un **GMENU** est similaire à un **TMENU** dans son fonctionnement, à ceci près qu'il crée un lien

sous forme d'image pour chaque page ou titre de navigation via la fonction TYPO3 interne du GIFBUILDER. Si ce type de menu est plus riche en termes de possibilités, il augmente cependant les temps de chargement de l'application, puisque tous les éléments de menu et leurs états doivent être inclus en tant qu'images.

Exemple : navigation secondaire dans « B2C »

En guise de deuxième illustration, un simple menu graphique est créé pour la sous-navigation dans le site « B2C » avec le gabarit `temp.subnavigation_html_gmenu`. Le premier niveau aligne les éléments de menu sur la gauche et trace deux lignes autour d'eux. Le second niveau est aligné à droite et comprend une ligne sur le côté droit. Les pages actives sont mises en couleurs (voir la figure 5.71).



Figure 5.71:
Sous-navigation sur
deux niveaux avec un
GMENU

Créez le cObject HMENU en imposant `entryLevel=1` et faites du premier niveau du menu un GMENU.

```
temp.subnavigation_html_gmenu = HMENU
temp.subnavigation_html_gmenu.entryLevel = 1
temp.subnavigation_html_gmenu.1 = GMENU
```

Précisez d'abord les propriétés de l'objet NO ; dans les objets GMENU, il s'agit, par définition, du type GIFBUILDER.

```
temp.subnavigation_html_gmenu.1 {
    NO {
```

Chaque élément de menu se termine par la balise `<br />`. La taille de l'image, `XY=150,40`, et la couleur de l'arrière-plan, `backColor=white`, sont définies par les propriétés de la fonction GIFBUILDER.

```
wrap = |<br />
XY = 150,40
backColor = white
```

Référence 023806

Les objets du GIFBUILDER sont créés via une liste numérique dans laquelle chaque élément est du type `GifBuilderObj`. De cette manière, l'élément de menu du premier niveau est composé de plusieurs objets.

Les objets 7 et 8 créent chacun un GifBuilderObj du type **BOX** et ceux-ci se chevauchent à cause de la hiérarchie.

```
# boîte grise
7 = BOX
7.dimensions = 16,18,120,20
7.color = #DFDFDF
# boîte blanche
8 = BOX
8.dimensions = 16,18,119,19
8.color = #FFFFFF
```

C'est parce que les propriétés **dimensions** ne diffèrent que d'un pixel que vous ne voyez que de fines lignes grises sur les bords de la boîte contenant le menu.

L'objet 10 est un GifBuilderObj de type **TEXT** et il contient le type de données **stdWrap**. La ligne de code **10.text.field=title** permet d'afficher le titre de la page. La propriété **fontFile** pointe vers la police de caractère TrueType utilisée dans la création d'images ; celle-ci doit d'abord être chargée dans le système de fichiers sous **fileadmin/...** La taille des polices de caractère et leur couleur sont définies dans les propriétés **fontSize** et **fontColor**. L'espacement est défini en coordonnées x/y par la propriété **offset** et l'alignement du texte par **align**.

```
10 = TEXT
10.text.field = title
10.fontFile = fileadmin/fonts/VERDANAB.TTF
10.fontSize = 11
10.fontColor = #FFFFFF
10.offset = 30,31
10.align = left
```

La propriété **niceText** qui affiche plus clairement les petites lettres ne fonctionne pas toujours comme prévu, selon la version d'ImageMagick que vous possédez ; elle mobilise également des ressources du serveur pour créer les images.

```
#10.niceText = 1
```

Le GifBuilderObj du type **SHADOW** permet de créer des ombres dont la position est fixée par les coordonnées de la propriété **offset**, tandis que le focus et la transparence sont déterminés par **blur** et **opacity**.

```
# Ombre pour le texte
10.shadow.offset = 1,1
10.shadow.blur = 80
10.shadow.opacity = 40
```

Le GifBuilderObj **TEXT** n'a pas de propriété **caractère gras**, mais vous pouvez simuler celle-ci en repassant le texte une seconde fois (**20<.10**) ou en imposant **iterations=2**.

```
# Simulation des caractères gras
20 < .10
}
```

Une autre possibilité, qui semble généralement meilleure, consiste à inclure une vraie police de caractère gras (10.fontFile=fileadmin/fonts/VERDANAB.TTF).

Dans l'état RO=1 (effet de survol) des éléments de menu, seule la couleur de la police de caractère est remplacée.

```
RO < .NO
RO {
  10.fontColor = #FFCC66
  20 >
  20 < .10
}
RO = 1
```

L'élément de menu à l'état actif ACT est associé à un autre GifBuilderObj BOX.

```
ACT < .RO
ACT {
  # Ombre
  5 = BOX
  5.dimensions = 18,20,120,20
  5.color = #D3D3D3
}
ACT = 1
}
```

Pour construire le menu de second niveau, on commence par l'initialiser en copiant toutes les données du menu de niveau 1.

```
temp.subnavigation_html_gmenu.2 = GMENU
temp.subnavigation_html_gmenu.2 < temp.subnavigation_html_gmenu.1
temp.subnavigation_html_gmenu.2 {
  NO {
```

La taille de l'image à créer est écrasée.

```
XY = 150,24
```

Les propriétés 7 et 8 (boîtes grise et blanche) sont supprimées.

```
7 >
8 >
```

Un nouveau GifBuilderObj de type BOX est créé dans l'objet 8 avec une largeur de 1 pixel et une hauteur de 24 pixels.

```
# ligne à droite
8 = BOX
8.dimensions = 136,0,1,24
8.color = #DFDFDF
```

Plusieurs propriétés du GifBuilderObj 10 de type TEXT sont écrasées.

```

10.offset = -25,16
10.fontSize = 10
10.align = right
20 < .10
}

```

L'état **RO** du niveau 2 reprend les propriétés de l'objet **NO** et écrase seulement la couleur de la police de caractère.

```

RO >
RO < .NO
RO {
    10.fontColor = #FFCC66
    20 >
    20 < .10
}
RO = 1

```

L'état actif **ACT** reprend les propriétés de l'objet **RO**.

```

ACT < .RO
ACT = 1
}

```

Le menu est à présent terminé. Intégrez le gabarit TS et l'objet temporaire dans le gabarit principal et vérifiez le résultat dans le frontend.

5.10.4 Menus basés sur des couches (TMENU_LAYERS/GMENU_LAYERS)

Référence 620108

Les objets de menu **TMENU_LAYERS** et **GMENU_LAYERS** se basent respectivement sur **TMENU** et **GMENU** tout en étendant leurs propriétés. Ils créent tous les deux un outil de navigation sur plusieurs niveaux via des couches DHTML et JavaScript. Pour cela, intégrez leurs fonctionnalités avec le script `media/scripts/tmenu_layers.php` et/ou `media/scripts/gmenu_layers.php`. Lorsque vous choisissez le menu, souvenez-vous du problème de la compatibilité des navigateurs. En particulier, les anciens navigateurs ne supportent pas les versions du *W3C Document Object Model* (DOM) ou les supportent seulement dans une certaine mesure.

Référence 599829

Les navigateurs à jour, tels qu'Opera 7, supportent les versions récentes du DOM. Si vous voulez utiliser un **TMENU_LAYERS**, il est judicieux de modifier les fonctions JavaScript correspondantes dans le fichier `/media/scripts/jsfunc.layermenu.js` (voir la référence ci-contre).

Exemple : navigation principale dans « B2C »

Pour la navigation principale dans le site « B2C », un menu à couche graphique est créé dans le gabarit `temp.navigation_html_gmenu_layers`.

Si vous survolez les éléments de menu du premier niveau, vous devriez voir la sélection des sous-pages dans une couche. Elle reste présente si une page du second niveau est active. Les icônes devant les éléments de menu et la couleur des polices de caractère changent lors du survol et lorsque l'état est actif.

Le résultat est visible à la figure suivante.



Figure 5.72:
Navigation principale
avec un
GMENU_LAYERS

Intégrez d'abord la bibliothèque `gmenu_layers.php` et créez ensuite un menu hiérarchique dans un objet temporaire (HMENU). Le premier niveau est un objet de menu `GMENU_LAYERS`.

```
includeLibs.gmenu_layers = media/scripts/gmenu_layers.php
temp.navigation_html_gmenu_layers = HMENU
temp.navigation_html_gmenu_layers.1 = GMENU_LAYERS
```

Déterminez à présent le comportement de la couche.

La propriété `layerStyle` permet d'assigner les attributs CSS à la balise `<div>` utilisée. Dans ce cas, la position en y est fixée à 195 pixels. `lockPosition` fixe la position des couches qui, dès lors, ne se réfère plus à la position de la souris. Choisissez x pour les menus horizontaux et y pour les menus verticaux. `xPosOffset` définit le décalage de la couche sur l'axe des x à partir de l'élément de menu. Cette propriété se réfère — selon que `lockPosition` est activée ou non — soit au coin supérieur gauche de chaque image du menu, soit à la position à partir de laquelle elle a été activée.

```
temp.navigation_html_gmenu_layers.1 {
  layerStyle = position:absolute;left:0;top:195px;width:10px;VISIBILITY:hidden;
  xPosOffset = -215
  lockPosition = x
```

La propriété `expAll=1` de l'objet de menu `GMENU` devrait être activée dans tous les cas, pour que le deuxième niveau puisse aussi bénéficier des effets de survol. Sinon, la navigation en couches ne fonctionnera pas.

```
expAll = 1
```

La propriété `displayActiveOnLoad` vous garantit que le menu en couches reste visible, même pour une page active du second niveau.

```
displayActiveOnLoad = 1
```

Le niveau 1 du menu est défini comme un `GMENU`, comme dans l'exemple de la navigation secondaire graphique dans le site « B2C ». D'abord nous activons l'état normal `NO` :

```
NO {
  backColor = white
```

La taille des éléments de menu est calculée automatiquement à partir de la longueur des images du tableau `GifBuilderObj`, dans ce cas, avec le propriété `XY` du `GIFBUILDER`. La donnée `x,y +calc` permet cet accès et fixe la valeur de X à `[10.w]+40` pixels et celle de Y à 30 pixels.

[10.w] représente la largeur w de l'objet 10 de la liste numérique, à laquelle nous ajoutons 40 pixels. De même, la hauteur de l'objet serait paramétrée par le code [10.h].

```
XY = [10.w]+40, 30
```

Les autres propriétés sont déjà connues. Deux GifBuilderObj du type BOX se chevauchant, 5 et 7, sont créés. L'objet 10 affiche le titre de la page et le met en forme.

```
5 = BOX
5.dimensions = 2,15,5,1
5.color = #666666

7 = BOX
7.dimensions = 4,13,1,5
7.color = #666666

10 = TEXT
10.text.field = title
10.text.case = lower
10.offset = 14,18
10.fontFile = fileadmin/fonts/ARIAL.TTF
10.fontSize = 12
10.fontColor = #666666
}
```

L'état RO est activé pour les effets de survol. Les propriétés de l'objet NO sont reprises et certaines valeurs de couleurs sont modifiées.

```
RO < .NO
RO {
  5.color = #E6B800
  7.color = #E6B800
  10.fontColor = #E6B800
}
RO = 1
```

Les éléments de menu, lorsqu'ils sont actifs, devraient être précédés d'un cadre jaune autour d'un signe moins. Pour ce faire, ajoutez deux carrés (un jaune et un blanc) qui se chevauchent à la liste numérique, sous la forme des GifBuilderObj 1 et 3 de type BOX, et supprimez l'objet 7 (le plus devient alors un moins).

```
ACT < .NO
ACT {
  # boîte autour du signe --> carré jaune ou blanc
  1 = BOX
  1.dimensions = 0,11,9,9
  1.color = #E6B800

  3 = BOX
  3.dimensions = 1,12,7,7
  3.color = white

  # non |
```

```

    7 >
  }
  ACT = 1
}

```

Le deuxième niveau du menu affiche les éléments de menu sous forme d'un **GMENU**. Cela n'apporte aucune nouvelle propriété : on ne l'utilise ici que dans un but pratique.

```

temp.navigation_html_gmenu_layers.2 = GMENU
temp.navigation_html_gmenu_layers.2.wrap = |<br />
temp.navigation_html_gmenu_layers.2 {
  NO {
    backColor = white
    XY = 120, 19

    # lignes de gauche et du dessous
    5 = BOX
    5.dimensions = 0,0,1,19
    5.color = #CCCCCC
    7 = BOX
    7.dimensions = 0,18,120,1
    7.color = #CCCCCC

    10 = TEXT
    10.fontFile = fileadmin/fonts/ARIAL.TTF
    10.fontSize = 11
    10.text.field = title
    10.text.case = lower
    10.offset = 12,10
    10.fontColor = #666666
  }

  RO < .NO
  RO.10.fontColor = #E6B800
  RO = 1

  ACT < .NO
  ACT.10.fontColor = #E6B800
  ACT = 1
}

```

Observez le menu dans le frontend, étendez les sous-menus pour les effets de survol et gardez-les ouverts pour les pages actives.

5.10.5 GMENU_FOLDOUT

L'objet de menu **GMENU_FOLDOUT** se limite à deux niveaux et génère une navigation dynamique via JavaScript. Pour le premier niveau du menu, utilisez un menu graphique et pour le second, choisissez soit un menu graphique, soit un menu de texte. La fonctionnalité est intégrée avec le script `media/script/gmenu_foldout.php`. N'oubliez pas que la compatibilité est restreinte pour les anciens navigateurs lorsque vous utilisez ce type de navigation.

Référence 623360

Exemple : sous-navigation « B2C »

En guise d'illustration, examinons la navigation secondaire du site « B2C », qui a jusqu'à présent été implémentée avec un TMENU. Créez un nouveau gabarit TS, `temp.subnavigation_html_gmenu_foldout`. Ceci devrait entraîner le déploiement dynamique des éléments de sous-menu du premier niveau lorsque vous cliquez dessus. La disposition est la même que celle du menu de texte et elle apparaît comme dans la figure suivante :

Figure 5.73:
Navigation
secondaire sur deux
niveaux avec un
`GMENU_FOLDOUT`



Les deux niveaux du menu sont construits avec l'objet de menu `GMENU`. Intégrez d'abord la bibliothèque des fonctions requises, `gmenu_foldout.php`, via `includeLibs`.

```
includeLibs.gmenu_foldout = media/scripts/gmenu_foldout.php
```

Définissez un `cObject` temporaire de type `HMENU`. La propriété `entryLevel=1` permet de démarrer le menu au premier niveau du site Web. Pour le premier niveau, nous définissons l'objet de menu `GMENU_FOLDOUT`.

```
temp.subnavigation_html_gmenu_foldout = HMENU
temp.subnavigation_html_gmenu_foldout.entryLevel = 1
temp.subnavigation_html_gmenu_foldout.1 = GMENU_FOLDOUT
```

Afin que les sous-pages du second niveau soient toujours affichées, vous devez activer la propriété `expAll` de l'objet de menu. `menuOffset` place le premier niveau du menu dans le coin supérieur gauche de la page. La propriété `subMenuOffset` permet de définir le décalage entre les sous-éléments du menu et l'élément principal correspondant. La couleur de l'arrière-plan est définie au moyen de la propriété `menuBackColor` et la hauteur et la largeur respectivement avec `menuWidth` et `menuHeight`.

```
temp.subnavigation_html_gmenu_foldout.1 {
  expAll = 1
  menuOffset = 17, 202
  subMenuOffset = 14, 25
  menuBackColor = white
  menuWidth = 164
  menuHeight = 800
```

Si vous voulez que le sous-menu se ferme lorsqu'un autre menu principal est appelé, fixez `stayFolded=0`. La vitesse d'ouverture dynamique du sous-menu est définie via le nombre d'étapes ; `foldSpeed=1` affiche immédiatement l'état final. Avec `displayActiveOnLoad=1`, les sous-menus s'ouvrent automatiquement si la page courante est située dans ce sous-menu.

```
stayFolded = 0
foldSpeed = 6
displayActiveOnLoad = 1
```

L'état normal **NO** est défini pour le premier niveau. La couleur de l'arrière-plan et sa taille sont spécifiées.

```
NO {
  backColor = #FFFFFF
  XY = 164, 22
```

L'objet 3 de la liste numérique insère une image sous la forme d'un `GifBuilderObj IMAGE` qui met en évidence les éléments de menu.

```
3 = IMAGE
3.file = fileadmin/images/layout/subnavi1_line.gif
3.offset = 3
```

L'objet 10 affiche le titre de la page et le met en forme.

```
10 = TEXT
10.text.field = title
10.text.case = lower
10.offset = 29,15
10.fontColor = #666666
10.fontFile = fileadmin/fonts/ARIAL.TTF
10.fontSize = 11
}
```

L'état actif **ACT** reprend de nouveau les propriétés de l'objet **NO** et ne change que quelques détails qui servent à afficher le titre de la page en majuscules (`case=upper`).

```
ACT < .NO
ACT {
  10.text.case = upper
}
ACT = 1
}
```

Le deuxième niveau du menu, `temp.subnavigation_html_gmenu_foldout.2`, est défini de la même manière. Contrairement au niveau 1, vous avez ici le choix entre un **TMENU** et un **GMENU**. Le **GMENU** a été utilisé dans cet exemple-ci, et ses états **RO** et **ACT** sont marqués par des icônes différentes.

```
temp.subnavigation_html_gmenu_foldout.2 = GMENU
temp.subnavigation_html_gmenu_foldout.2 {
  NO {
```



```

XY = 150,18
backColor = white

3 = IMAGE
3.file = fileadmin/images/layout/subnavi2_line.gif
3.offset = 5

10 = TEXT
10.text.field = title
10.text.case = lower
10.offset = 24,13
10.fontColor = #666666
10.fontFile = fileadmin/fonts/ARIAL.TTF
10.fontSize = 11
}
RO < .NO
RO {
    12 = IMAGE
    12.file = fileadmin/images/icons/subnavi2_ro.gif
    12.offset = 15,2
}
RO = 1

ACT < .RO
ACT {
    12.file = fileadmin/images/icons/subnavi2_act.gif
}
ACT = 1
}

```

Insérez le gabarit TS et l'objet temporaire dans le gabarit principal du site « B2C » et vérifiez le résultat dans le frontend. Pour éviter que chaque page ne se recharge à chaque ouverture du sous-menu, créez des pages dans le premier niveau de votre application qui admettent des sous-pages sans contenu, comme le type de pages **Shortcut**.

La propriété `dontLinkIfSubmenu` ouvre le second niveau du menu s'il a des sous-pages, mais n'affiche pas cette page sous forme de lien. Le processus de chargement est omis.

```

...
dontLinkIfSubmenu = 1

```

Les pages sans sous-pages sont affichées comme d'habitude avec leur lien.

5.10.6 ImageMaps (IMGMENU)

Référence 610406

L'objet de menu **IMGMENU** génère un menu à partir de l'image en arrière-plan et des titres des pages du niveau de menu à afficher. Pour les éléments de menu, des zones sensibles sont configurées avec des balises `<area>`, créant des liens vers les pages correspondantes. Les éléments de menu sont des objets **GIFBUILDER** que vous pouvez structurer comme vous l'entendez. Pour les états **NO**, **ACT**, **SPC**, etc., **GIFBUILDER** crée des images différentes dans chaque cas.

Exemple : navigation principale dans « B2C »

Pour le site « B2C », une nouvelle navigation principale peut à présent être implémentée avec l'objet de menu **IMGMENU** via un nouveau gabarit TS, **temp.navigation_imgmenu**. Le résultat suivant est alors généré dans le frontend.



Figure 5.74:
Navigation principale
avec un **IMGMENU**

Le menu représente l'outil de navigation principale, c'est pourquoi il démarre au niveau Root-level : **entryLevel=0**. La propriété **maxItems** précise que six éléments de menu au maximum peuvent s'afficher.

```
temp.topmenu = HMENU
temp.topmenu.entryLevel = 0
temp.topmenu.maxItems = 6
```

Le menu n'admet qu'un niveau, un **IMGMENU**. Spécifiez l'image de l'arrière-plan sur laquelle est affiché le titre au moyen de la propriété **main** du type de données **GIFBUILDER**.

Comme nous l'avons déjà montré, les objets du **GIFBUILDER** sont générés sous la forme d'une liste numérique dont chaque élément a le type de données **GifBuilderObj**. L'objet **10** crée un **GifBuilderObj** de type **IMAGE**, qui lie l'image voulue avec la propriété **file**.

Référence 023805

```
temp.topmenu.1 = IMGMENU
temp.topmenu.1 {
  main.10 = IMAGE
  main.10.file = fileadmin/images/layout/bg_imgmenu.gif
```

La taille originale de l'image est entrée dans les propriétés **XY** et la palette de couleurs de l'image est réduite à 16 couleurs via **reduceColors**, ce qui a pour effet de réduire la taille du fichier.

```
main.XY = [10.w], [10.h]
main.reduceColors = 16
```

Définissez le point de départ du menu à partir du coin supérieur gauche de l'image avec **dWorkArea**.

```
dWorkArea = 12,22
```

L'état normal du menu est défini. La propriété **distrib** spécifie la distance ente les éléments de menu. L'objet **10** affiche le titre de la page comme un **GifBuilderObj TEXT** et définit sa présentation.

```
NO {
  distrib = textX+25, 0
  10 = TEXT
  10.text.field = title
```

```

10.fontSize = 12
10.fontColor = #666666
10.fontFile = fileadmin/fonts/ARIAL.TTF
10.niceText = 1
10.offset = 2,0

```

imgMap vous permet d'accéder à une propriété de GIFBUILDER qui crée une carte d'images (imagemap) pour le fichier image à partir du GifBuilderObj TEXT. La propriété **explode** élargit la zone sensible avec les coordonnées x/y.

```

10.imgMap.explode = 3,2

```

Le signe « plus » qui précède chaque élément de menu est mis en pratique via deux GifBuilderObj de type BOX.

```

# |
20 = BOX
20.dimensions = -5,-6,1,5
20.color = #666666
# -
30 = BOX
30.dimensions = -7,-4,5,1
30.color = #666666
}

```

L'état actif ACT reprend toutes les propriétés de l'objet NO. L'objet 20 de la liste numérique est supprimé (« plus » devient « moins »), et un cadre est créé avec les objets 40, 50, 60 et 70.

```

ACT < .NO
ACT {
  20 >
  #- en dessous
  40 = BOX
  40.dimensions = -9,0,9,1
  40.color = #666666
  # - au-dessus
  50 < .40
  50.dimensions = -9,-8,9,1
  # | à gauche
  60 < .40
  60.dimensions = -9,-8,1,9
  # | à droite
  70 < .40
  70.dimensions = -1,-8,1,9
}
ACT = 1
}

```

Le menu est terminé. Intégrez le gabarit TS **temp.navigation_imgmenu** et l'objet temporaire **temp.topmenu** dans le gabarit principal et testez le résultat dans le frontend.

5.10.7 Menus JavaScript (JSMENU)

L'élément de menu JSMENU configure une navigation pour laquelle les pages sont sélectionnées via deux listes déroulantes de menus qui dépendent les unes des autres. Chaque menu de sélection représente un niveau du site Web. La caractéristique la plus importante de ce menu par rapport à un simple menu de sélection est que les niveaux les plus bas sont repris avec les entrées correspondantes qui dépendent du plus haut niveau sélectionné, sans que la page ne doive être rechargée dans le navigateur. Un JSMENU peut afficher jusqu'à cinq niveaux (cinq listes déroulantes de menus).

Référence 421175

Exemple : navigation principale dans « B2B »

Dans cet exemple, nous modifions la navigation principale du site « B2B » avec le gabarit TS temp.navigation_tswrap_jsmenu. La figure suivante montre le menu dans son état final lorsque la sous-page « Produit 1 » est sélectionnée.

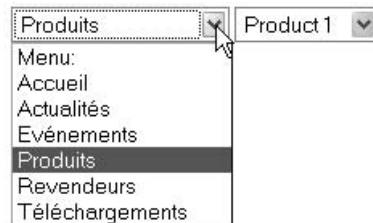


Figure 5.75:
Le menu principal en
JSMENU

Définissez l'objet temporaire temp.jsmenu_tswrap comme un cObject HMENU et le premier niveau du menu comme un JSMENU.

```
temp.jsmenu_tswrap = HMENU
temp.jsmenu_tswrap.1 = JSMENU
```

Le nombre de niveaux à afficher est déterminé à l'aide de la propriété levels=2.

```
temp.jsmenu_tswrap.1 {
  levels = 2
```

La liste numérique vous permet de définir le comportement de différents niveaux. La liste déroulante des menus de niveau 1 est séparée de la liste déroulante pour le niveau suivant par un séparateur « espace vide ». Si vous voulez sélectionner à l'avance la page active, alors posez la valeur showActive=1. Si aucune page n'est sélectionnée, alors le premier élément de la liste déroulante n'a normalement pas de valeur. Dans ce cas, vous pouvez donner un nom à cet élément grâce à la propriété firstLabel.

```
1 {
  wrap = |&nbsp;
  showActive = 1
  firstLabel = Menu:
}
```

Les valeurs du premier niveau passent au deuxième niveau et seul leur nom est écrasé.

```
2 < .1
2.firstLabel = Sous-menu:
}
```

Le menu est terminé : vous pouvez l'inclure dans le gabarit principal du site « B2B ».

5.10.8 Menus .special

Référence 743914

Le cObject HMENU affiche par défaut la structure de la page sous la forme d'une arborescence, mais il peut générer d'autres structures grâce à la propriété **special**. Par exemple, un menu rootline affiche les pages du rootline. Une configuration simple ressemble à ceci :

```
temp.navigation = HMENU
temp.navigation.entryLevel = 0
temp.navigation.special = rootline
temp.navigation {
    1 = TMENU
    1.NO.allWrap = |/&nbsp;
}
```

Ce code produit le résultat suivant :

```
B2C-Accueil/ Produits/ Produit 2/
```

La propriété **special** détermine la structure particulière que le HMENU doit créer. Les types suivants sont disponibles. Les propriétés de chaque type de menu se trouvent dans le document TSref.

directory

Le type **directory** génère un menu à partir des sous-pages de l'ID des pages spécifiées ou à partir de la page courante.

list

list vous permet de créer un menu pour des pages précises déterminées par leur ID.

updated

Si vous voulez afficher des pages selon qu'elles ont été mises à jour ou non, utilisez **updated**.

Rootline

Le chemin de la page racine à la page active aide d'habitude l'utilisateur à garder une vue d'ensemble des applications complexes. Ce menu rootline est généré par le type **rootline**.

browse

Le type **browse** vous permet de naviguer à travers les pages de différentes manières (précédent, suivant, ...)

keywords

Le type **keywords** crée un menu de pages qui ont un ou plusieurs mots-clés en commun avec la page courante dans l'en-tête de page. Cela facilite l'affichage des pages interdépendantes.

userdefined

Le type `userdefined` sert à contrôler les menus avec votre propre script PHP. La référence `TSref` contient un exemple détaillé de ce menu.

Exemple : directory

Presque chaque application comprend des pages ne devant pas apparaître dans le contexte de la navigation principale. Dans l'application exemple « BT3 », ceci s'applique aux éléments de menu méta « Accueil », « Impression », « Plan de site » et « Contact ». Ils doivent toujours être présents et sont stockés ici dans le Dossier Système **Fonctions** de l'arborescence des pages. Utilisez le type de menu `directory` pour l'affichage.

Le menu doit être structuré comme sur la figure suivante, et placé en dessous de la navigation secondaire, sur la gauche de la page.



Figure 5.76:
Information méta
créée par la propriété
`special.directory`

Créez votre propre gabarit TS `temp.metas_html` pour le site « B2C » et définissez un `HMENU` comme objet temporaire. La propriété `special` reçoit le type de menu désiré avec la valeur `directory`. La propriété `value` reprend les ID des pages auxquelles le menu se réfère. Puisque la valeur fait référence aux ID des pages du Dossier Système **Fonctions**, c'est une constante définie en conséquence qui est assignée à la valeur.

```
temp.metas_html = HMENU
temp.metas_html.special = directory
temp.metas_html.special.value = {$metas.pid}
```

Vous pouvez configurer le menu comme vous l'entendez ; dans l'exemple, il est configuré en `TMENU` et est présenté sous forme de tableau.

```
temp.metas_html.1 = TMENU
temp.metas_html.1 {
  wrap = <br /><br /><br /><table width="170" cellspacing="0" cellpadding="0" border="0">|</table><br /><br /><br />
  NO {
    linkWrap = <font face="Arial,Helvetica,sans-serif,sans-serif" size="2" color="#666666" class="metas">|</font>
    allWrap = <tr><td></td><td align="left" valign="top">|</td></tr>
    ATagBeforeWrap = 1
```

Les images précédant le titre des pages sont affichées grâce à la commande `import.field=media` (le champ **Fichiers** dans l'en-tête des pages correspondantes).

```

    # obtenez l'image de la page (le champ Fichiers des pages de type Avancé)
    beforeImg.import = uploads/media/
    beforeImg.import.field = media
    beforeImg.import.listNum = 0
  }
}

```

Pour les sites « B2B » et « B2C » basés sur des balises <div>, le même menu est réalisé par le gabarit TS `temp.metas_autoparser_tswrap` :

```

temp.metas_autoparser_tswrap = HMENU
temp.metas_autoparser_tswrap.special = directory
temp.metas_autoparser_tswrap.special.value = {$metas.pid}
temp.metas_autoparser_tswrap.1 = TMENU
temp.metas_autoparser_tswrap.1 {
  noBlur = 1
  wrap = <ul id="metas">|</ul>
  NO {
    wrapItemAndSub = <li>|</li>
    beforeImg.import = uploads/media/
    beforeImg.import.field = media
    beforeImg.import.listNum = 0
  }
}

```

À présent, insérez la navigation méta dans le gabarit principal adéquat avec **Include basis template** et remplacez les balises spécifiques en copiant les objets temporaires correspondants. Utilisez le cObject COA pour l'intégration, puisque la navigation secondaire et l'information méta doivent s'afficher groupées dans une sous-partie ou dans une balise <div>.

Exemple « B2C » :

```

page.10 {
  ...
  subparts.LEFT = COA
  subparts.LEFT.10 < temp.subnavigation_html
  subparts.LEFT.20 < temp.metas_html

```

Exemple « B2B »/« B2E » :

```

page {
  ...
  # Gauche / menu secondaire
  40 = COA
  40.10 < temp.subnavigation_autoparser_tswrap
  40.20 < temp.metas_autoparser_tswrap
  40.stdWrap.wrap = <div id="subnavigation">|</div>

```

Exemple : rootline

Un menu rootline est prévu dans l'en-tête de l'application exemple. Il montre à l'utilisateur le chemin courant. Les titres des pages apparaissent sous forme de liens. En pratique, cela devrait ressembler à la figure suivante :

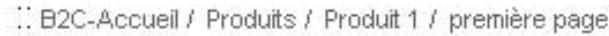


Figure 5.77:
Le menu rootline
indique le chemin de
la page courante

Définissez en premier lieu un **HMENU** avec la propriété `special=rootline`.

```
temp.rootline_html= HMENU
temp.rootline_html.special = rootline
```

La propriété **range** vous permet de préciser les niveaux de début et de fin. Si vous omettez la valeur de fin (ou si vous la fixez à -1), le chemin complet est affiché. Avec -2, la page courante ne sera pas incluse dans le chemin, avec -3, les deux derniers niveaux ne sont pas repris, etc. Notez que vous ne pouvez pas utiliser d'espace dans la valeur.

```
temp.rootline_html.special.range = 0|-2
```

Dans l'exemple, le menu est un **TMENU** basé sur du texte. La propriété **wrap** insère deux fois le caractère « deux points » devant les éléments de menu. **linkwrap** fournit une balise **** à l'état normal **NO**.

```
temp.Rootline_html.1 = TMENU
temp.Rootline_html.1 {
    wrap = &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&::|
    target = _top
    NO {
        linkWrap = <font face="Arial,Helvetica,sans-serif,sans-serif" size="1" color="#666666" class="Rootline">|</font>
```

Les éléments de menu sont ici séparés par une barre oblique, sauf pour le dernier élément de menu. Pour insérer une barre oblique à la fin du menu, utilisez `allWrap=.`

La solution est d'utiliser la fonction `optionSplit` qui est disponible pour une série de propriétés et de valeurs des objets de menu. `optionSplit` est utilisée principalement lorsque vous devez fixer des valeurs pour toute une série d'éléments. Pour les menus, la même valeur est normalement définie pour plusieurs éléments de menu. Si vous utilisez `optionSplit`, la valeur est analysée par la fonction `et`, selon la définition, des valeurs différentes sont assignées aux éléments. Il est intéressant à ce niveau-ci de remarquer que trois valeurs séparées par la syntaxe `|*` sont présentes dans la propriété `allWrap`. Le dernier élément de menu n'est pas terminé par une barre oblique. La syntaxe et les règles de priorité de cette fonction sont expliquées plus loin, à la section 5.11.1.

Référence 819809

[illegible]

Pour finir, voici le menu rootline à partir du gabarit `TypoScript temp.Rootline_autoparser_tswrap` pour les sites « B2B » et « B2E ». Les éléments de menu apparaissent sous la forme d'une liste numérique et sont mis en forme avec CSS.


```

temp.rootline_autoparser_tswrap = HMENU
temp.rootline_autoparser_tswrap.special = rootline
temp.rootline_autoparser_tswrap.special.range = 0|-2
temp.rootline_autoparser_tswrap.1 = TMENU
temp.rootline_autoparser_tswrap.1 {
    wrap = <ul>::&nbsp;  |</ul>
    noBlur = 1
    NO {
        linkWrap= <li>|</li>
        # optionSplit: "/" après chaque élément sauf le dernier
        allWrap = |*| |&nbsp;  /&nbsp;  |*| |*|
    }
}

```

Insérez les menus rootline et vérifiez le résultat dans le frontend.

À ce stade de la construction, l'application « BT3 » est presque complète. Tous les menus sont configurés et intégrés. Seules quelques fonctionnalités manquent comme par exemple la version pour l'impression. Celles-ci seront ajoutées au chapitre suivant. En outre, d'autres cObjects y seront introduits, et les fonctions seront décrites plus en détail.

5.11 TypeScript en détail

5.11.1 La fonction optionSplit

Référence 819808

La fonction `optionSplit` est disponible dans les propriétés TS dont le type de données porte le même nom. Elle est souvent utilisée dans les objets de menu. Cette fonction permet d'assigner des valeurs différentes à des groupes d'éléments faisant partie d'une série.

La syntaxe

Les règles de base sont simples mais deviennent plus complexes lorsqu'elles sont combinées.

|*|

Répartit les valeurs dans les zones principales, respectivement les zones *Première*, *Médiane* et *Dernière*.

||

Divise les zones principales en trois sous-zones au maximum : *première*, *deuxième* et *troisième*.

- Les valeurs sont assignées dans l'ordre suivant : d'abord *Dernière*, puis *Première* et enfin *Médiane*.
- Si la valeur *Médiane* est vide (" "), la dernière partie de la *Première* valeur est répétée.
- Si la *Première* valeur et la valeur *Médiane* sont vides, la première partie de la *Dernière* valeur est répétée avant la *Dernière* valeur.
- Les valeurs de la zone principale *Médiane* sont répétées.

Exemple : COLUMNS

Le cObject **COLUMNS** permet de créer un tableau à plusieurs colonnes avec du contenu. Cet objet admet entre autres la propriété **gapBgCol** du type de données **HTML-color / stdWrap +optionSplit**. Elle détermine une couleur d'arrière-plan pour les interstices entre les cellules de la table. En accédant à la fonction **optionSplit**, vous pouvez spécifier des couleurs différentes pour certains interstices.

COLUMNS:			
Property:	Data type:	Description:	Default:
tableParams	<TABLE>-params		border=0
gapBgCol	HTML-color / stdWrap +optionSplit	background-color for the gap-tablecells	

Figure 5.78:
Grâce au type de données, vous vérifiez si *optionSplit* est disponible.

Nous configurons rapidement ci-dessous une table à sept colonnes, donc avec six interstices. Les propriétés **tableParams** et **totalWidth** fixent respectivement les paramètres de la balise **<table>** et la largeur, ici à 500 pixels. L'épaisseur des interstices et des lignes entre les colonnes est précisée dans **gapWidth** et **gapLineThickness** et le nombre de colonnes est fixé dans **rows**. Enfin, nous assignons du contenu aux différentes colonnes.

```
10 = COLUMNS
10 {
  tableParams = cellspacing="0" cellpadding="0" border="0"
  totalWidth = 500
  gapWidth = 30
  gapLineThickness = 1
  rows = 3
  gapBgCol = red
  1 = TEXT
  1.value = colonne1
  ...
  7 = TEXT
  7.value = colonne7
}
```

Nous voudrions attirer l'attention ici sur une propriété intéressante qui utilise **optionSplit**, à savoir **gapBgCol**, qui a reçu uniquement la valeur **red** jusqu'à présent : tous les interstices sont donc rouges (nous avons mis en évidence les couleurs dans les figures suivantes).

colonne1 ■■■ colonne2 ■■■ colonne3 ■■■ colonne4 ■■■ colonne5 ■■■ colonne6 ■■■ colonne7

Figure 5.79:
Valeur : red

Lorsque vous spécifiez trois valeurs (une pour chaque zone principale) avec **optionSplit** :

```
gapBgCol = red |*| green |*| yellow
```

alors le *premier* espace est rouge, le *dernier* jaune et tous les autres prennent la même couleur que l'espace *médian* (règle 4).

Figure 5.80:

Valeur : red |*| green
|*| yellow



La répétition de la valeur *médiane* (règle 4) est mieux perçue si la valeur est divisée en sous-zones.

```
gapBgCol = red |*| green || fuchsia |*| yellow
```

Ici, nous avons choisi les couleurs verte et fuchsia pour la zone *médiane* et elles seront donc affichées en alternance.

Figure 5.81:

Valeur : red |*| green
|| fuchsia |*| yellow



Les zones principales *Première* et *Dernière* sont à présent subdivisées en plusieurs valeurs. La *Première* zone reçoit trois valeurs, la zone *Médiane* une et la *Dernière* deux.

```
gapBgCol = red || aqua || grey |*| green |*| yellow || fuchsia
```

Tous les interstices sont représentés dans leur couleur correspondante :

Figure 5.82:

Valeur : red || aqua ||
grey |*| green |*|
yellow || fuchsia



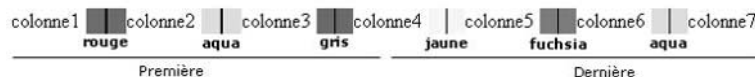
Fixez ensuite trois valeurs pour les zones principales *Première* et *Dernière* (ou réduisez le nombre de colonnes avec la propriété *rows*) :

```
gapBgCol = red || aqua || grey |*| green |*| yellow || fuchsia || aqua
```

Les zones principales sont traitées dans l'ordre défini précédemment pour déterminer les couleurs à afficher ; en effet, nous avons spécifié sept couleurs pour six interstices. D'abord les valeurs de la zone principale *Dernière* sont traitées, ensuite celles de *Première* et enfin celles de *Médiane* (règle 1). La valeur de la zone principale *Médiane* n'est plus prise en compte, étant donné que tous les interstices ont déjà été définis par les zones *Dernière* et *Première*.

Figure 5.83:

aqua || grey |*| green
|*| yellow || fuchsia ||
aqua



Si aucune valeur n'est assignée à la zone principale *Médiane*,

```
gapBgCol = red || grey |*||*| yellow
```

la dernière valeur de la zone principale *Première* est répétée (règle 2). Notez qu'il ne devrait pas y avoir d'espace entre les séparateurs. Dans l'exemple, la valeur « grey » est répétée.



Figure 5.84:
Valeur : red || grey
|*|*| yellow

Si les zones principales *Première* et *Médiane* ne contiennent pas de valeurs, la première valeur de la zone principale *Dernière* est répétée (règle 3).

```
gapBgCol = |*|*| yellow || fuchsia || aqua
```

Dans l'exemple, les valeurs de la zone principale *Dernière* sont fixées en premier. Les interstices restants prennent la couleur jaune, comme la première valeur de la zone principale *Dernière*.

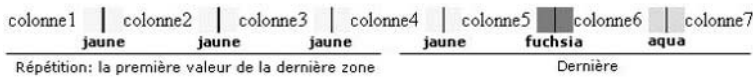


Figure 5.85:
Valeur : |*|*| yellow
|| fuchsia || aqua

Nous avons déjà utilisé la fonction `optionSplit` pour les menus rootline des gabarits TS `temp.rootline_html` et `temp.rootline_autoparser_tswrap`. Une caractéristique particulière de cette fonction est qu'elle est appliquée à tous les objets du GIFBUILDER avant que les éléments de menu ne soient créés. Il en va de même pour toutes les propriétés `allWrap` utilisées.

```
temp.rootline_autoparser_tswrap.1.NO {
    ...
    # optionSplit: "/" après chaque élément sauf le dernier
    allWrap = |*| |&nbsp;   |&nbsp;   |*| |*|
    ...
}
```

La propriété `allWrap` reçoit trois valeurs :

- La *Première* zone est vide `|*|`.
- La zone *Médiane* contient une valeur qui se répète, à savoir l'enveloppe `|    |    |*|`.
- La *Dernière* zone contient un espace vide : c'est pourquoi il y a un dernier séparateur `|*|`. Sinon, les espaces vides au début et à la fin de la chaîne de caractères seraient supprimés, conformément aux règles de TypoScript.

La fonction `optionSplit` peut sembler plutôt complexe à première vue, mais elle est aussi très puissante, étant donné que vous pouvez assigner différentes valeurs à une série d'éléments sans aucune programmation supplémentaire. Pour les menus, `optionSplit` est un outil dont on se passe difficilement.

5.11.2 Travailler avec des images et le GIFBUILDER

Si TYPO3 est installé avec les bibliothèques graphiques recommandées, vous pouvez manipuler les images avec la fonction `GIFBUILDER`. Les menus graphiques en particulier les utilisent et créent automatiquement des images en format GIF à partir des éléments de menu. Mais ils peuvent aussi produire des images à partir d'en-têtes combinés avec des polices

Référence 788778

de caractère. C'est de cette manière que la disposition 5 des en-têtes pour les éléments de contenu est créée, sur base du gabarit standard **content (default)** qui intègre le gabarit de base **styles.header.gfx1**. L'élément de contenu **Textbox** fait aussi appel au **GIFBUILDER**.

Objets GIFBUILDER (GifBuilderObj)

Le **GIFBUILDER** comprend une série de propriétés générales, par exemple **XY**, **offset** ou **reduceColors**, pour définir respectivement la taille de l'image, un décalage général pour tous les objets à partir du coin supérieur gauche de l'image, et la réduction du nombre de couleurs de l'image GIF. Une zone de travail est créée dans l'image à l'aide de la propriété **workArea**. Les objets courants sont générés dans une liste numérique de **GifBuilderObj**. Soyez attentifs : dans TSref, les références à des objets sont fixées exclusivement pour les **GifBuilderObj** et non pour les **cObjects** du même nom, comme **TEXT** et **IMAGE**.

Les objets **GIFBUILDER** suivants peuvent être utilisés directement ou via d'autres **GifBuilderObj** :

TEXT

Le **GifBuilderObj TEXT** crée une image à partir de texte. Via le type de données **stdWrap** et **getText**, vous pouvez y faire référence séparément.

SHADOW

En tant que **GifBuilderObj**, **SHADOW** est une propriété de **TEXT** qui crée un effet d'ombre. Si elle est utilisée seule, vous devez créer un lien à partir de l'objet **TEXT** auquel elle se réfère, via **textObjNum**.

EMBOSS

Les mêmes remarques s'appliquent au **GifBuilderObj EMOSS**. Il crée deux copies décalées derrière l'objet **TEXT** afin de produire une sorte de relief. En utilisant les propriétés, vous pouvez leur donner différentes couleurs, les rendre plus douces ou plus transparentes.

OUTLINE

Le **GifBuilderObj OUTLINE** est aussi une propriété de **TEXT**. Il renforce les contours et est seulement influencé par la valeur de la couleur et l'intensité. Le résultat n'est en général pas très bon. Il est préférable d'utiliser une ombre avec une forte intensité.

BOX

BOX crée un rectangle dont vous précisez la taille et l'alignement.

IMAGE

Le **GifBuilderObj IMAGE** du type de données **imgResource** est associé à une image. Vous pouvez même utiliser la fonction **GIFBUILDER** via laquelle tous les **GifBuilderObj** sont disponibles.

EFFECT

Avec **EFFECT**, il est possible d'effectuer une rotation sur l'image, de la renverser, de fixer les nuances de gris, de changer les valeurs gamma, etc. Les valeurs sont ajoutées avec la propriété **value** et séparées avec la commande **tube**, « | ».

WORKAREA

Si vous insérez une nouvelle **WORKAREA** dans la liste numérique des **GifBuilderObj**,

les objets suivants feront référence à la nouvelle zone de travail (par exemple avec la propriété `offset`).

CROP

CROP restreint l'affichage de l'image à des zones partielles. Ce `GifBuilderObj` adapte la `workArea` aux nouvelles dimensions de l'image.

SCALE

SCALE met les images à l'échelle grâce aux propriétés `width` et `height`. Ce `GifBuilderObj` adapte également `workArea` aux nouvelles dimensions de l'image.

ADJUST

ADJUST vous permet de corriger les valeurs des tons de l'image en définissant les valeurs des tons gris d'entrée et de sortie. Si vous devez corriger les valeurs gamma, utilisez le `GifBuilderObj EFFECT`. Les propriétés sont reprises dans une liste via `value` et séparées par le signe tube, « | ».

[IMGMAP]

IMGMAP n'est pas un `GifBuilderObj`, mais il est utilisé conjointement avec `TEXT` pour créer une zone sensible (imagemap) pour un fichier GIF. Il est utilisé pour l'objet de menu `IMGMENU`.

+calc

Si vous ajoutez `+calc` au type de données d'une propriété du `TSref`, vous pourrez faire des opérations arithmétiques sur la valeur. Les opérateurs utilisés, `+`, `-`, `/` et `*` ne suivent pas de règles de priorité et sont traités dans l'ordre dans lequel ils sont écrits. Toutefois, pour le `GIFBUILDER`, vous pouvez aussi vous référer aux valeurs d'autres objets. Cela permet par exemple de calculer la longueur et la largeur de la figure globale en fonction de `GifBuilderObj` comme `TEXT` ou `IMAGE`.

```
...
XY = [100.w]+160 , [100.h]+5
```

Pour les dimensions de l'image, cela signifie :

```
XY = [largeur du GifBuilderObj 100]+160 Pixel , [hauteur du GifBuilderObj
100]+5 Pixel
```

.niceText

La bibliothèque *Freetype* intégrée dans `GDLib` et utilisée par `PHP` ne supporte pas de manière satisfaisante l'anti-aliasing dans toutes les versions. La propriété `niceText` aide à afficher plus clairement les petites lettres. Elle fournit une solution de rechange en demandant à `TYPO3` de doubler la taille du texte sur un masque, avant de réduire l'échelle jusqu'à la taille correcte. Ce faisant, la fonction `ImageMagick combine` est utilisée pour placer (pour masquer) le texte dans l'arrière-plan. Cette procédure mobilise beaucoup de puissance de la part du serveur pour créer les images, mais procure un meilleur résultat selon la version d'`ImageMagick` utilisée. Puisque les images ne sont pas recrées à chaque appel, mais qu'elles le sont seulement après

le premier appel qui suit l'effacement du cache, vous devez décider à quel endroit vous voulez utiliser `niceText`. Notez que cette propriété peut produire différents résultats selon les versions de GDlib, Freetype et ImageMagick utilisées, de sorte que votre menu graphique sur le serveur en ligne peut avoir une autre apparence que celui sur le serveur de production.

Exemple : l'image dans l'en-tête

Référence 016166

À la section « Étapes de construction avec des gabarits en cascade », à la page 281, une image a déjà été insérée dans la zone d'en-tête du frontend. Le type d'objet **IMAGE** utilisé accepte deux méthodes d'intégration d'images, grâce à la propriété `file` du type de données `imgResource`.

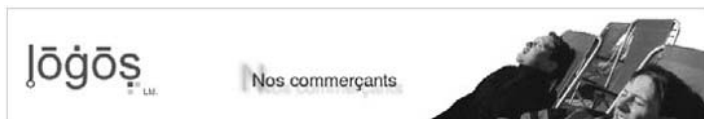
- Il fait référence à un fichier dans le système de fichiers via le champ **Fichiers** dans l'en-tête de la page.
- Il utilise le **GIFBUILDER**, une fonction qui vous permet de créer une image qui combine plusieurs objets graphiques tels que des images, du texte ou des boîtes.

La première méthode a été illustrée dans le gabarit TS `temp.header_html`.

L'utilisation de la fonction **GIFBUILDER** est illustrée par le gabarit TS `temp.GIFBUILDER_header_image`. Si vous voulez suivre et reconstruire l'exemple, vous le trouverez dans le Dossier **Système Gabarits Principaux/cObjects**.

L'en-tête ne doit pas seulement comprendre l'image incluse dans les en-têtes de page, comme c'était le cas jusqu'ici, mais aussi inclure le titre de la page courante sous forme de graphique. Le texte est ombragé et la première lettre est mise en couleur.

Figure 5.86:
Inclusion du
graphique comme un
objet **GIFBUILDER**



D'abord, créez un objet temporaire défini comme un cObject **IMAGE**. Par la propriété `file` du type de données `imgResource`, une nouvelle image est créée avec la fonction **GIFBUILDER**.

```
temp.header_image_gifbuilder = IMAGE
temp.header_image_gifbuilder.file = GIFBUILDER
```

La taille de l'image est déterminée avec **XY**.

```
temp.header_image_gifbuilder.file {
    XY = 588,125
```

L'objet **10** de la liste numérique du **GifBuilderObj** est du type **IMAGE** et prend en argument les images assignées à partir du champ **Fichiers** de l'en-tête de page. Le paramètre `slide` lance une recherche dans l'arborescence des pages à partir de la page courante jusqu'au niveau 1. Cette recherche s'arrête dès qu'une image a été trouvée.

```
10 = IMAGE
10.file.import = uploads/media/
10.file.import.data = levelmedia:1, slide
```

Déterminez la position de l'objet **10** avec la propriété `offset` qui prend comme référence le coin supérieur gauche de l'image globale (alternativement, si `workArea` est activé, c'est la zone de travail qui est prise en référence).

```
10.offset = 199,0
```

Ensuite, la première lettre jaune du titre de la page est créée. Définissez l'objet **18** comme un `GifBuilderObj` de type `TEXT`. Sa propriété `text` est du type de données `stdWrap` et essaie d'abord de lire le sous-titre de la page via `text.field = subtitle`. S'il n'y en a pas, c'est le titre qui est lu.

```
18 = TEXT
18.text.field = subtitle // title
```

Les propriétés `case` et `crop` précisent respectivement que le texte sera écrit en majuscules et que seule la première lettre sera affichée.

```
18.text.case = upper
18.text.crop = 1
```

L'objet **18** est placé dans la figure globale avec décalage ; il reçoit une taille et une couleur pour la police de caractère.

```
18.offset = 62,90
18.fontSize = 40
18.fontColor = #FFCC33
```

La propriété `niceText` améliore la reproduction des polices de caractère.

```
18.niceText = 1
```

Le `GifBuilderObj TEXT` accède au `GifBuilderObj EMBOSS` via la propriété `emboss`. Les deux copies de la lettre sont décalées d'une valeur spécifiée par `offset` ; `blur` les adoucit et `opacity` en contrôle la transparence.

```
18.emboss {
  highColor = #FFCC33
  lowColor = #FFCC33
  offset = 2,2
  blur = 99
  opacity = 50
}
```

De même, le texte entier est généré dans l'objet **20**.

```
20.text.field = subtitle // title
20.offset = 75,85
20.fontSize = 20
20.fontColor = #333333
20.niceText = 1
```

Ici, le `GifBuilderObj TEXT` accède au `GifBuilderObj SHADOW` via `shadow` et produit une ombre argentée. Elle est en décalage par rapport au coin supérieur gauche de l'objet **20**, via `offset`. Elle est adoucie par `blur`, est mise à un niveau de 60% de transparence par `opacity`, et son intensité est définie par `intensity`.

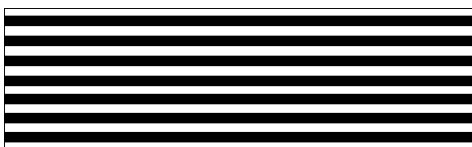

```

20.shadow {
    color = silver
    offset = 7,8
    blur = 70
    opacity = 60
    intensity = 15
}

```

Bien sûr, vous pouvez étendre l'exemple comme vous l'entendez. Avec la propriété `mask` du `GifBuilderObj IMAGE`, un masque noir et blanc est appliqué sur l'image.

Figure 5.87:
Masque placé sur
l'image



Ajoutez la ligne suivante à la configuration TypeScript :

```
10.mask = fileadmin/mediapool/material_intern/gifbuilder-mask.png
```

Seules les parties de l'image couvertes par les zones blanches du masque sont visibles pour l'objet `GIFBUILDER 10`. Les zones noires sont donc invisibles.

Figure 5.88:
L'image de l'en-tête
avec le masque



Comme variante, vous pouvez ajouter un nouvel objet `GIFBUILDER` (l'objet 15). C'est un `Gif-BuilderObj` du type `IMAGE` qui crée une image avec l'aide de la fonction `GIFBUILDER`. La propriété `XY` permet de préciser à nouveau la taille de l'image et l'objet `GIFBUILDER 100` prend en argument le chemin vers l'image dans le système de fichiers.

Figure 5.89:
L'image présente le
logo TYPO3 sur un
fond gris



```

15 = IMAGE
15.file = GIFBUILDER
15.file {
    XY = [100.w]+160 , [100.h]+5
    100 = IMAGE
    100.file = fileadmin/mediapool/material_intern/typo3logo.gif

```

La propriété `offset` permet de déterminer la position du logo dans notre image.

```
100.offset = 160, 5
```

L'arrière-plan de l'image est rendu transparent avec `transparentBackground = 1`. Le même résultat est obtenu via `transparentColor = #CCCCCC` avec la valeur de la couleur grise du logo.

```
transparentBackground = 1
transparentColor = #CCCCCC
}
```

Le logo TYPO3 est donc parfaitement intégré à l'image de l'en-tête.



Figure 5.90:
Combinaison de
plusieurs images

Vous pouvez remarquer qu'il n'y a pas de limites à l'utilisation de la fonction `GIFBUILDER`. En fin de compte, la limite, c'est votre créativité.

5.11.3 La fonction `stdWrap`

La fonction `stdWrap` joue un rôle central dans TypeScript et reflète aussi le concept de TS. Nous l'avons déjà beaucoup utilisée dans les exemples. Cette fonction s'est développée à partir du concept d'enveloppe pour pallier au fait que TypeScript ne peut pas manipuler les structures de contrôle (sauf pour les `conditions`), puisque ce n'est pas un véritable langage de programmation.

`stdWrap`, une sorte de « canif suisse », propose au développeur TypeScript une série de propriétés et de fonctions pour fournir des données, changer ou éditer des éléments de contenu ou encore pour permettre de traiter des conditions.

Les propriétés et les fonctions de `stdWrap` sont utilisables partout où une propriété d'objet du type de données `stdWrap` est définie. Les objets ont aussi fréquemment une propriété nommée `stdWrap` qui rend disponible cette fonction.

Selon leurs finalités, les propriétés de `stdWrap` sont séparées en trois zones : **Get data**, **Override/Conditions** et **Parse data**. Notez que les propriétés sont traitées dans l'ordre dans lequel elles sont reprises dans `TSref`.

Get data

La zone **Get data** des propriétés est utilisée pour rendre disponibles les données ou pour les publier. Cela peut servir pour le titre d'une page, la date courante, un `cObject`, un nombre de lignes dans une requête de base de données, une liste de fichiers dans un répertoire ou une variable globale. Les propriétés les plus importantes sont `data`, `field`, `current`, `cObject` et `filelist`.

data

Affiche, via la syntaxe `Type:pointeur`, les valeurs de différents tableaux PHP comme, par exemple, des enregistrements de données.

Exemple : ... `.data = page:title`

field

Renvoie généralement sous forme de valeur les éléments de contenu d'un champ de la base de données à partir de l'enregistrement courant de données.

Exemple : ... `.field = title`

current

Détermine, par le biais d'une valeur booléenne, l'accès au contenu actuel du registre des données. Dans TYPO3, le registre des données est un tampon interne où sont stockées les valeurs courantes pour certaines fonctions. Les variables enregistrées à cet endroit sont disponibles globalement et sont publiées, sauveées ou écrasées à différents moments du traitement.

cObject

Charge le contenu d'un objet de contenu.

Exemple : ... `.cObject = IMAGE`

filelist

Lit un répertoire de fichiers et fournit une liste de fichiers. Les paramètres suivants sont séparés par le signe `|` :

1. Chemin
2. Liste délimitée par des virgules (sans espace) des types de fichiers possibles (gif, jpg, ...)
3. L'ordre du tri : `name, size, ext, date`
4. `r` inverse l'ordre du tri
5. Si activé (non vide), la liste de fichiers est créée avec le chemin complet et donc pas uniquement avec le nom du fichier

Exemple : ... `.filelist = fileadmin/img/|jpg,png|name||1`

Une aide supplémentaire est fournie par la publication dans le frontend de la liste de l'enregistrement courant des données, via la propriété `stdWrap debugData`.

\$cObj->data:

\$cObj->data:	
uid	47
pid	18
tstamp	1087374477
sorting	128
deleted	0
perms_userid	1
perms_groupid	1
perms_user	31
perms_group	27
perms_everybody	0
crdate	1081438117
cruser_id	1
title	Home
doktype	2
TSconfig	
treeStop	0
author	
author_email	
nav_title	
content_from_pid	0
mount_pid	0
alias	
nav_hide	0
mount_pid_ol	0
currentValue_kidjls9dksoje	index.php?id=18

Figure 5.91:
Affichage de
l'enregistrement
courant de données
dans le frontend

Override/Conditions

Les propriétés dans cette zone sont utilisées pour écraser et comparer des valeurs. Vous pouvez, entre autres, fixer une nouvelle valeur qui dépend d'autres valeurs. Les propriétés les plus importantes sont :

override

Écrase la valeur d'un objet si la valeur de **override** n'est pas vide « » ou 0. La question de savoir si l'objet original a déjà une valeur assignée ou s'il est vide n'est pas pertinente dans ce cas.

ifEmpty

Assigne une nouvelle valeur à un objet si cette valeur est vide jusqu'à présent « ». Le zéro est ici traité comme une valeur.

listNum

Extrait la valeur désirée des éléments de contenu qui sont séparés par une virgule.

trim

Supprime les espaces vides, les tabulations et les sauts à la ligne.

required

Si la valeur courante est vide « », l'objet faisant l'appel à **stdWrap** est retourné en l'état. Les propriétés **stdWrap** qui suivent ne sont pas prises en considération.

if

Inclut la fonction **if** du même nom. Est utile pour vérifier si une ou plusieurs conditions sont satisfaites dans le but de fixer certaines valeurs.

fieldRequired

Si le champ de l'enregistrement actuel de données est vide, la valeur courante est supprimée.

Parse data

La zone **Parse data** détermine des propriétés pour le traitement des données. Toute une série d'opérations sont disponibles ici. Le contenu HTML peut être filtré par la fonction **HTMLparser**, et certaines balises supprimées. Avec **split**, les éléments de contenu sont divisés en plusieurs valeurs sur base de caractéristiques précises avant d'être traités. D'autres fonctions qui sont incluses via des propriétés sont **encapsLines**, **addParams**, **textStyle**, **tableStyle**, **filelink** et **typolink**.

Get data et Parse data

field et **data** ont déjà été utilisées dans les exemples précédents en tant que propriétés **stdWrap** de la zone **Get data**.

field

Avec **field**, le sous-titre ou le titre des pages

```
...
NO {
    10 = TEXT
    10.text.field = subtitle // title
```

ou encore une image du champ **Fichiers** de l'en-tête de page sont lus pour être insérés dans les menus.

```
NO {
    # Lire l'image de la page
    beforeImg.import = uploads/media/
    beforeImg.import.field = media
```

data

La propriété **data** est utilisée pour sélectionner l'image, non seulement dans la page courante, mais aussi pour toutes les pages du rootline, jusqu'au moment où une image est trouvée à un niveau supérieur.

```
temp.header_image = IMAGE
temp.header_image.file.import = uploads/media/
temp.header_image.file.import.data = levelmedia:1, slide
```

Les propriétés de l'objet `stdWrap` de la zone `Parse data` ont déjà servi dans les premières étapes de la section 5.2. Cet exemple simple affiche le texte en majuscules à l'aide de la propriété `case` :

```
page.10 = HTML
page.10.value = kasper
page.10.value.case = upper
```

Exemple : version pour l'impression

Dans l'application exemple, la version pour l'impression manque. Vous devriez rajouter un bouton qui optimise la page courante en vue de son impression. La mise en page n'est pas prise en compte dans la version pour l'impression, seul le contenu est mis en forme suivant les styles CSS. La page est appelée avec le paramètre URL `&type=98` qui correspond en interne au nombre `typeNum=98` réservé pour l'impression. Afin de créer le lien vers l'aperçu avant impression, utilisez les propriétés des zones `Get data` et `Parse data` de l'objet `stdWrap`.



Figure 5.92:
Le bouton
d'impression appelle
la page avec le
paramètre `&type=98`

Créez un nouveau gabarit TS. Dans les exemples, vous le trouverez sous l'appellation `temp.printversion` dans le Dossier Système `Gabarits Principaux/Fonctions`. Le bouton d'impression est généré et nous définissons une page qui restituera le résultat correspondant.

Créez d'abord un cObject temporaire de type COA.

```
temp.printversion = COA
temp.printversion {
```

Le bouton est créé avec la propriété `wrap` appliquée à l'ensemble de l'objet.

```
wrap = <a href="|" name="Printversion" title="Printversion" target="_blank" class="printversion">$printlabel</a>
```

Toutefois, l'URL contenant le paramètre `&type=98` n'est pas encore reliée au bouton. Dans ce but, la propriété `10` de la liste numérique du COA a été définie comme un cObject `TEXT`.

```
10 = TEXT
```

Les propriétés `stdWrap`, qui sont des outils du cObject `TEXT`, sont directement appliquées à la base de l'objet, ce qui ne correspond pas au comportement standard des autres objets. Afin d'obtenir l'URL de la page courante, c'est la propriété `data` qui est utilisée.

La propriété est du type de données `getText`, ce qui signifie qu'elle peut afficher les valeurs de différents tableaux, accessibles par le système en utilisant la syntaxe `Type:pointeur`.

Référence 293098

L'adresse courante est déterminée par le type `getIndpEnv` et le pointeur `REQUEST_URI`.

```
10.data = getIndpEnv:REQUEST_URI
```

Le tableau contient dès lors une valeur telle que `/index.php?id=18`. Elle peut être encore traitée par la propriété `wrap` afin d'ajouter le paramètre `&type=98`.

```
10.wrap = |&type=98
```

Référence 463307

Notez qu'il s'agit ici d'un simple exemple de création du lien vers les versions d'impression. Si vous utilisez l'option `TS config.simulateStaticDocuments`, cet exemple et le suivant, ne fonctionneront pas — ces derniers sont uniquement destinés à mettre en évidence les propriétés `stdWrap`. Une extension permettant de créer des liens correctement est disponible dans le TER (Typo3 Extension Repository).

Une autre possibilité est d'utiliser la propriété `split`. Elle vous permet de diviser la valeur en plusieurs parties via les propriétés de la fonction du même nom⁹.

`10.split.token=/` détermine la chaîne de caractères à utiliser comme séparateur. `/index.php?id=18` est divisé en deux valeurs. La première est vide et la seconde contient la chaîne de caractères qui suit le séparateur : `index.php?id=18`. L'objet `10.split.cObjNum` du type de données `cObjNum` +`optionSplit` sert de pointeur vers les `cObjects` d'une liste numérique.

Ces objets (1, 2, 3, ...) sont appelés pour traiter les valeurs séparées par `optionSplit`. `cObjNum` détermine quels objets sont appelés pour chaque valeur séparée. Si `cObjNum=1`, chaque valeur séparée est passée à l'objet 1 pour le traitement.

Vous pouvez aussi utiliser `optionSplit` pour traiter chaque valeur séparée par un objet différent. Le mode de fonctionnement exact de `optionSplit` est expliqué à la section 5.11.1. Dans la configuration suivante, l'objet 2, qui ajoute le paramètre `type` à partir de la valeur séparée avec `current=1`, est appelé uniquement pour la dernière valeur (`index.php?id=18`). Toutes les autres valeurs séparées sont traitées par l'objet 1, qui fixe une valeur vide avec `override=`.

```
/* Variante 2: traitement par la fonction split
10.split.token = /
10.split.cObjNum = 1 |*| 1 |*| 2
10.split.1.override =
10.split.2.current = 1
10.split.2.wrap = |&type=98
*/
}
```

Le résultat est identique à celui de la première variante, qui est plus courte.

Vous devez à présent définir le gabarit pour l'affichage de la page dans son format d'impression avec `typeNum=98`. La sélection de gabarits par l'utilisation de `type/typeNum` a été vue à la section 5.8.

⁹Ceci n'a pas de sens dans ce contexte, puisqu'une seule valeur est traitée ; nous l'utilisons ici uniquement pour expliciter la fonction `split`

```
# un autre gabarit de page pour le type 98 (=version impression)
alt_print = PAGE
alt_print {
    typeNum = 98
```

La CSS suivante détermine le format de l'affichage.

```
CSS_inlineStyle = body, p, h1, h2, h3, h4 { font-family: Arial, Verdana, sans-serif; font-size: 11pt; } h1, h2, h3, h4 { font-size: 12pt; font-weight: bold; }
```

Pour finir, le contenu de la colonne **Normal** est publié.

```
10 < styles.content.get
}
```

Override/Conditions

Comme nous l'avons fait remarquer au début, il est possible de lier les propriétés à des conditions. Les propriétés `stdWrap` qui écrasent les valeurs ou fixent les conditions ont déjà servi dans les exemples.

listNum

Pour le menu `temp.metas_html` qui reprend l'information méta, la première image du tableau a été paramétrée avec la propriété `listNum=0` lorsque les images étaient lues à partir de l'entête de page. Même s'il y a plusieurs images dans le champ Fichiers, seule celle-ci est affichée.

```
NO {
    beforeImg.import = uploads/media/
    beforeImg.import.field = media
    beforeImg.import.listNum = 0
```

ifEmpty

Pour la mise en page de base du gabarit TS `ts CTABLE template`, un espace vide (` `) a été passé en argument dans la propriété `ifEmpty` pour que le tableau soit toujours affiché même s'il ne possède pas de contenu.

```
page.30 = CTABLE
page.30 {
    ...
    c.10 = COA
    c.10.5 = HTML
    c.10.5.value = <table border="0" cellspacing="0" cellpadding="0" width=
"586"><tr>
    c.10.10 < styles.content.get
    c.10.10.wrap = <td align="left" valign="top">|<br /><br /></td>
    c.10.25 = HTML
    c.10.25.value = </tr></table>
    c.10.stdWrap.ifEmpty = &nbsp;
```


Exemple : utilisation de deux colonnes

Jusqu'à présent, seul le contenu de la colonne **Normal** était affiché dans les gabarits principaux. Il n'y a que le gabarit TS `ts wrap template` qui était inclus dans la colonne **Right** de la disposition de base. Une solution pour insérer du contenu dans une deuxième colonne est la suivante : créer deux gabarits TS différents inclus dans l'arborescence des pages et les utiliser selon les besoins. Puisque certaines pages ne sont pas publiées en deux colonnes, nous avons besoin d'une solution plus flexible, qui crée la zone de droite dans le gabarit seulement si les éléments de contenu existent vraiment.

Nous allons présenter cette solution à l'aide du gabarit TS `auto parser template`. Pour ce faire, nous créons plusieurs objets dans le chemin d'objet `temp.*`. D'abord, l'objet `styles.content.get` est copié dans `temp.inhalt` pour rechercher les éléments de contenu de la table `tt_content` dans la colonne **Normal** (`colPos=0`).

```
# contenu de gauche
temp.inhalt < styles.content.get
```

Le second objet temporaire, `temp.rechts`, affiche les éléments de contenu de la colonne **Droite** (`colPos=2`) avec l'objet `styles.content.getRight`.

```
### contenu de droite ###
temp.right < styles.content.getRight
```

La propriété `stdWrap.required` force l'arrêt du traitement de `stdWrap` si l'objet n'a produit aucun contenu : dans ce cas, le contenu de la colonne de droite. S'il y a du contenu dans la colonne de droite, il est entouré par une balise `<div>` pour le disposer sur la page en utilisant la propriété `stdWrap innerWrap`.

```
temp.right.stdWrap.required = 1
temp.right.stdWrap.innerWrap = <div id="right"><div id="content_right">|
</div><div id="rightfooter"></div> </div>
```

Le contenu de la colonne **Normal** est placé devant cela avec la propriété `preCObject` et est enveloppé d'une balise `<div>`.

```
# placement du contenu de gauche en tant que cObject avant le contenu de droite
temp.right.stdWrap.preCObject < styles.content.get
temp.right.stdWrap.preCObject.wrap = <div id="content_left"> | </div>
```

La configuration `temp.right` définit un objet qui affiche les éléments de contenu des colonnes **Normal** et **Droite**, pourvu que du contenu soit présent dans la colonne de droite ; sinon, rien n'est affiché. La colonne **Normal** est paramétrée ici avec l'ID `content_left` au lieu de `content` dans la balise `<tag>`, ce qui signifie que la colonne peut être configurée plus finement par CSS que pour un affichage en une seule colonne.

L'objet `temp.inhalt` est remplacé par `temp.right` s'il y a du contenu dans la colonne de droite. On réalise cela avec la propriété `stdWrap override`.

La propriété `override` est elle-même du type de données `string/stdWrap` et peut donc afficher une chaîne de caractères ou des données via les propriétés du groupe de fonctions `Get Data` de `stdWrap`.

```
temp.inhalt.stdWrap.override.cObject < temp.right
```

Le contenu de l'objet `temp.inhalt` est finalement copié dans la sous-partie.

```
temp.mainTemplate = TEMPLATE
temp.mainTemplate {
    ...
    subparts.content < temp.inhalt
}
```

La mise en page qui en résulte comporte deux zones de contenu s'il y a du contenu dans la colonne Droite.



Figure 5.93:
Affichage dans le
frontend

Vous voyez que la fonction `stdWrap` est un outil puissant pour lire le contenu, pour construire des structures de contrôle de comparaison, et enfin pour traiter le contenu. S'il vous manque des fonctionnalités, vous avez la possibilité de traiter le contenu avec vos propres fonctions PHP, via les propriétés `preUserFunc` et `postUserFunc`.

5.11.4 Conditions

Nous avons déjà introduit les conditions. Elles doivent être satisfaites pour que le code TS qui les suit soit pris en compte. Elles constituent une autre structure de contrôle. Rappelez-vous que si plusieurs conditions sont vérifiées, seule la première condition satisfaite est prise en compte. Les conditions se terminent par `[END]` ou `[GLOBAL]`. Elles ne peuvent apparaître à l'intérieur d'un opérateur `{ }` qui représente l'emboîtement de propriétés. La condition `[ELSE]` est forcément remplie si la condition qui la précède ne l'est pas.

Référence 315080

Vous pouvez aussi combiner les conditions, comme dans l'exemple suivant, où le code TS est traité uniquement si le système appelant peut s'identifier comme étant un système Linux, ou si son IP est du type 145.153.102.*.

```
[system = linux] [IP = 145.153.102.*]
...
[END]
```

Nous n'avons pas encore utilisé de conditions dans nos exemples. Nous commençons donc par présenter les conditions reprises dans la dernière version de TSref avant d'indiquer par un exemple comment on les met en pratique.

Vue d'ensemble

Référence 501292

La vue d'ensemble qui suit reprend toutes les conditions que vous pouvez utiliser. Les valeurs exactes se trouvent dans TSref.

browser

Syntaxe : [browser =navigateur1, navigateur2,...]

[browser = msie] s'applique à toutes les versions de MS Internet Explorer

[browser = msie, opera] s'applique à tous les navigateurs MS Internet Explorer et Opera

[browser = opera7] s'applique à tous les navigateurs Opera 7.xx dont, par exemple, Opera 7.1.

Notez que les navigateurs peuvent être identifiés sous d'autres noms, de sorte que la requête ne mène pas alors au résultat désiré.

version

Syntaxe : [version =value1 ,>value2 ,=value3 ,<value4,...]

Cette condition concerne la version du navigateur. Les valeurs sont des nombres à virgule flottante avec « . » comme séparateur décimal. Les trois opérateurs suivants s'appliquent à la valeur de la version : = La valeur doit correspondre exactement. [version = 5.5]

> La numéro de la version doit être plus grand que la valeur précisée. [version = > 5.5]

< Le numéro de la version doit être plus petit que la valeur précisée. [version = < 5.5]

system

Syntaxe : [system=system1,system2]

La chaîne de caractères est comparée à la première partie de l'identification du système d'exploitation, et on considère que la condition est vraie s'ils correspondent.

[system = win9] correspond à Win95 et Win98.

[system=win,mac] correspond aux systèmes d'exploitation Windows et Mac.

device

Syntaxe : [device=device1,device2]

La condition est vraie si la chaîne de caractères correspond au matériel accédant au site (pda, wap, grabber, robot).

useragent

Syntaxe : [useragent=agent]

Pour vérifier s'il y a une correspondance avec les variables `getenv("HTTP_USER_AGENT")`.

L'astérisque * peut être utilisée au début et/ou à la fin de la chaîne de caractères.

[useragent=Lotus-Notes/4.5(Windows-NT)] correspond à HTTP_USER_AGENT « Lotus-Notes/4.5(Windows-NT) ».

[useragent = Lotus-Notes*] y correspond également.

language

Syntaxe : [language =lang1,lang2,...]

La variable `getenv("HTTP_ACCEPT_LANGUAGE")` est comparée avec une valeur donnée ; la condition est vraie si la correspondance est vérifiée dans son ensemble.

IP

Syntaxe : [IP =ipaddress1,ipaddress2,...]

La variable `getenv("REMOTE_ADDR")` est comparée à la valeur donnée. Cette dernière peut contenir le symbole * ou consister en une, deux ou trois parties :

[IP = 145.*.*] correspond à toutes les adresses IP qui commencent par 145.

hostname

Syntaxe : [hostname =hostname1,hostname2,...]

La valeur qui s'écrit sous forme d'une liste de noms de domaines, séparés par des virgules, doit correspondre à la variable `getenv("REMOTE_HOST")`. Le symbole * est permis, mais ne peut être utilisé dans une chaîne de caractères.

[hostname = votredomaine*.com] est correct.

[hostname = votredomaine*.com] est erroné.

hour

Syntaxe : [hour =heure1,>heure2,<heure3,...]

Les valeurs (0-23), séparées par des virgules, sont comparées à l'heure sur le serveur.

Les opérateurs possibles sont =, > et <.

minute

Syntaxe : [minute =...]

La minute spécifiée (0-59) est comparée à la minute sur le serveur.

dayofweek

Syntaxe : [dayofweek =...]

Le jour de la semaine (de 0/dimanche à 6/samedi) est comparé au jour de la semaine sur le serveur et la condition renvoie « vrai » si les deux correspondent.

dayofmonth

Syntaxe : [dayofmonth =...]

Le jour du mois (1 à 31) est comparé au jour sur le serveur.

month

Syntaxe : [month =...]

Les mois (de 1/janvier à 12/décembre) sont comparés au mois sur le serveur.

usergroup

Syntaxe : [usergroup =groupe1-uid,groupe2-uid,...]

La condition est satisfaite si l'utilisateur identifié dans le frontend est un membre du groupe d'utilisateurs spécifié. Le symbole * couvre tous les groupes d'utilisateurs configurés, spécifiés par la variable globale `gr_list`.

loginUser

Syntaxe : `[loginUser =fe_users-uid, fe_users-uid,...]`

L'uid d'un utilisateur frontend identifié est comparé avec le nombre entier donné. Le symbole * est utilisé pour savoir si l'utilisateur est bel et bien identifié.

treeLevel

Syntaxe : `[treeLevel =niveau1,niveau2,...]`

Si l'un des nombres entiers donnés correspond au niveau actuel dans le rootline, alors la condition est remplie. 0 correspond au Rootlevel, 1 correspond au premier niveau de menu.

`[treeLevel = 2,3]`

PIDinRootline

Syntaxe : `[PIDinRootline =pages-uid,pages-uid,...]`

La condition est satisfaite si la page actuelle correspond à l'un des pid spécifiés ou si l'une de ses sous-pages est impliquée. C'est seulement dans ces cas que le code TypoScript qui suit est exécuté. Si vous souhaitez assigner une valeur de couleur différente aux en-têtes dans différentes zones de l'arborescence des pages, utilisez le code suivant :

```
[PIDinRootline = 37]
content.wrap.header1 = <h1 class="bleu">|</h1>
[END]
[PIDinRootline = 16]
content.wrap.header1 = <h1 class="rouge">|</h1>
[GLOBAL]
```

L'en-tête de la page 37 et ses sous-pages sont en bleu, la page 16 et ses sous-pages en rouge.

PIDupinRootline

Syntaxe : `[PIDupinRootline =pages-uid,pages-uid,...]`

Voir `PIDinRootline`. La différence réside en ce que l'ID de la page actuelle n'est pas repris dans la comparaison.

globalVar

Syntaxe : `[globalVar=var1=valeur,var2<valeur2,var3>valeur3,...]`

La condition est remplie si la valeur de la variable correspond à la valeur des variables système. Les variables sont séparées par des virgules.

Les opérateurs possibles sont : >, <.

`[globalVar = GP:L=3]` s'applique par exemple à l'URL `...index.php?id=45&L=3`.

globalString

Syntaxe : `[globalString=var1=valeur,var2=*valeur2,var3=*valeur3*,...]`

Idem que `globalVar` sauf que les valeurs sont comparées en tant que chaînes de caractères. Le symbole * peut être placé au début ou à la fin.

`[globalString = HTTP_HOST=www.monsite.com]` et `[globalString = HTTP_HOST=*.monsite.com]` correspondent tous les deux à la valeur `http://www.monsite.com`.

userFunc

Syntaxe : [userFunc =user_match(checkLocalIP)]

Avec **userFunc**, vous utilisez vos propres fonctions PHP pour la vérification.

Exemple : étendre la version pour l'impression

Le bouton que nous avons utilisé jusqu'à présent dans le gabarit TS **temp.printversion** pour appeler la version pour l'impression fonctionne très bien, mais uniquement tant que vous ne créez pas d'adresses URL statiques avec l'option TS **config.simulateStaticDocuments**.

C'est pourquoi le gabarit est étendu ci-après. La chaîne de caractères est éditée et reçoit différents paramètres selon le type d'URL.

Pour l'objet existant **temp.***, l'objet **10** est d'abord reconfiguré pour les adresses URL statiques.

```
temp.printversion = COA
temp.printversion {
    wrap = <a href="|" name="Printversion" title="Printversion" target=
    "_blank" class="printversion">$printlabel</a>
    # simulateStatic Version, tous les liens inclus
    # seront aussi traités.
    10 = TEXT
    10.data = getIndpEnv:REQUEST_URI
```

L'URL affichée via la propriété **stdWrap** est divisée en chaînes de caractères par la propriété **listNum.splitChar** et sa valeur « . ». En posant **listNum=0**, seul le premier élément de la valeur est pris en compte et il est enveloppé par **|.98.html**. (Le tout fonctionne si vous avez inclus les paramètres avec **config.simulateStaticDocuments_pEnc_onlyP** en tant que valeur de hachage MD5 lorsque vous avez créé l'adresse URL statique.)

```
10.listNum.splitChar = .
10.listNum = 0
10.wrap = |.98.html
}
```

L'objet **temp.printversion.10** déjà existant est soumis à une **Condition** et n'est traité que si celle-ci est vérifiée.

globalString vérifie si la variable du serveur **ENV:REQUEST_URI** contient la valeur **/index.php***.

Le symbole ***** sert à vérifier la chaîne de caractères juste pour la valeur de départ **/index.php**.

```
# version "normale", pour les pages contenant index.php, avec les paramètres si
# nécessaire, par exemple : ?id=3434&L=1
[globalString = ENV:REQUEST_URI = /index.php*]
```

Si la condition est remplie, l'objet actuel **temp.printversion.10** est supprimé et redéfini, comme dans la version précédente.

```
temp.printversion.10 >
temp.printversion {
    10 = TEXT
    10.data = getIndpEnv:REQUEST_URI
    10.wrap = |&type=98
}
```

La condition se termine avec [global].

```
[global]
...
```

Le gabarit de la page reste inchangé. La version pour l'impression peut à présent être utilisée dans les deux versions.

Exemple : sélection de la langue

TYPO3 convient très bien aux applications multilingues. Nous avons déjà montré comment le rédacteur s'acquitte dans le backend de la maintenance du contenu en plusieurs langues. Celles-ci devraient maintenant pouvoir être sélectionnées dans le frontend via un lien qui appelle la page par le paramètre `&L=[sys_language_uid]`, lien devant lequel est placé un petit drapeau représentant la langue à choisir.

Figure 5.94:
Changement de
langue : « Français »
est actif, « English »
peut être choisi (à
gauche) et vice versa
(à droite)



Ceci est mis en pratique via un script PHP, mais peut se faire plus simplement en TypoScript.

Dans le `setup` du nouveau gabarit TS `temp.language`, le français est défini comme étant la langue par défaut (`sys_language_uid=0`) via la propriété `language=fr` du TLO `config`.

```
config.sys_language_uid = 0
config.language = fr
```

Si l'URL contient le paramètre `L=1`, la langue dont l'ID est 1 (`sys_language_uid=1`), l'anglais dans notre cas, est définie comme étant la deuxième langue. Puisqu'il ne s'agit pas d'une définition globale, la donnée `GET/POST` est vérifiée par la condition `[globalVar=GP:L=1]`.

```
# Langue anglaise, sys_language.uid = 1
[globalVar = GP:L = 1]
config.sys_language_uid = 1
config.language = en
[global]
```

Le lien effectif est défini comme un objet temporaire via le cObject `COA`. L'objet `10` de type d'objet `TEXT` fournit une langue alternative qui dépend de la langue courante. C'est pourquoi `value` est une constante. Un tableau qui met en forme l'affichage est placé autour de l'ensemble de l'objet avec la propriété `stdWrap outerWrap`. Cette dernière contient le drapeau de la langue alternative, qui est aussi une constante.

```
temp.language = COA
temp.language {
  10 = TEXT
  10.value = {$languageVersion}
  10.outerWrap = <table width="160px" bgcolor="#FFFFFF" border="0" cellpadding="1"
cellspacing="0"><tr><td width="12px"></td><td width="16px"></td><td width="170px">|</td></tr></table>
```

La propriété `stdWrap typolink` permet d'afficher l'objet 10 sous forme de lien. La propriété `parameter` détermine l'ID actuel. Avec `additionalParams`, la troisième constante est assignée au paramètre additionnel `&L`. `AtagParams` ajoute une classe CSS à la balise `<a>`.

```
10.typolink {
  parameter.data = page:alias // TSFE:id
  additionalParams = &L={$foreignLanguageID}
  ATagParams = class="lang"
}
}
```

Il ne vous reste plus qu'à définir les constantes exactes pour le lien, en fonction de la langue choisie. Si celle-ci est la langue par défaut, le français (`[globalVar=GP:L=0]`), aucune condition n'est requise et le paramètre du lien est configuré en fixant `foreignLanguageID` à 1. Le texte et l'icône du drapeau sont spécifiés respectivement par `languageVersion` et par `flagSmall`.

Constantes :

```
foreignLanguageID = 1
languageVersion = English version
flagSmall = fileadmin/images/icons/flaguk.gif
```

Si la langue sélectionnée est différente de la langue par défaut, le français, c'est-à-dire que le paramètre `L` de l'URL courante a une valeur `>0` (`[globalVar=GP:L>0]`), alors ajoutez une condition avec `globalVar` dans le champ `constants` et redéfinissez les constantes.

```
[globalVar = GP:L>0]
foreignLanguageID = 0
languageVersion = Version française
flagSmall = fileadmin/images/icons/flagfr.gif
[global]
```

Insérez le changement de langue dans le gabarit principal et vérifiez le résultat dans le front-end. Vous verrez que tout fonctionne uniquement avec du code TypoScript.

5.12 Travailler avec des cadres

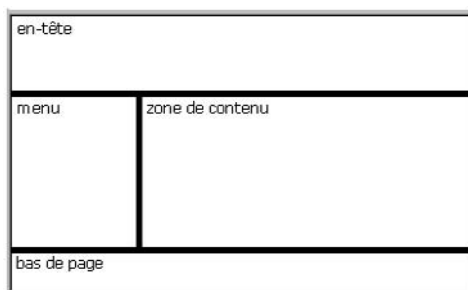
Même si travailler avec des cadres n'est plus considéré comme étant la meilleure pratique, nous désirons, par souci d'exhaustivité, vous montrer comment construire avec TYPO3 des sites basés sur des cadres.

À l'aide de cadres, vous pouvez diviser le champ visuel de votre navigateur en différents segments que vous définissez librement. Chaque segment forme des pages distinctes d'une application Web et peut en appeler d'autres via des liens et l'attribut `target="nom du cadre"`. Un fichier HTML comprenant le code suivant agence les pages `top.htm`, `menu.htm`, `content.htm` et `bottom.htm` dans un jeu de cadres.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Le titre</title>
</head>
<frameset rows="205,*,50" framespacing="4" frameborder="1" bordercolor="
#000000">
  <frame src="top.htm" name="top" frameborder="0" scrolling="no" marginwidth="0"
marginheight="0">
  <frameset cols="195,*">
    <frame src="menu.htm" name="menu" frameborder="0" scrolling="no" marginwidt
h="0" marginheight="0">
    <frame src="content.htm" name="content" frameborder="0" scrolling="no" marg
inwidth="0" marginheight="0">
  </frameset>
  <frame src="bottom.htm" name="bottom" frameborder="0" scrolling="no" marginwid
th="0" marginheight="0">
</frameset>
</html>
```

Si la page est appelée par le navigateur, elle affichera clairement les cadres. Le contenu est fourni par des pages particulières.

Figure 5.95:
Résultat de l'exemple
HTML dans le
navigateur



5.12.1 Création de cadres

Référence 234280

Pour créer un site avec des cadres en utilisant TYPO3, vous devez définir quelques pages dans un gabarit TS.

```
myframeset = PAGE
top = PAGE
menu = PAGE
content = PAGE
bottom = PAGE
```

Chaque page reçoit une valeur différente via la propriété `typeNum`.

```
myframeset.typeNum = 0
top.typeNum = 1
menu.typeNum = 2
content.typeNum = 3
bottom.typeNum = 4
```

Un objet de contenu de type `TEXT` est créé pour chaque page, excepté `myframeset`.

```
top.10 = TEXT
top.10.value = en-tête
menu.10 = TEXT
menu.10.value = menu
content.10 = TEXT
content.10.value = contenu
bottom.10 = TEXT
bottom.10.value = bas de page
```

Vous pouvez déjà appeler les pages individuellement dans le navigateur via l'`id` et le paramètre `type` correspondant. Le concept de sélection de gabarit avec `type/typeNum` a déjà été présenté à la section 5.8.

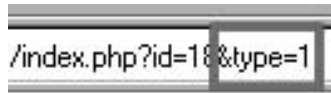


Figure 5.96:
Affichage de pages
avec le paramètre
`type`

Si vous appelez une page sans le paramètre `type`, ce dernier sera fixé à la valeur par défaut (0) et appellera donc le gabarit `myframeset`. Nous définissons à présent ce dernier comme un jeu de cadres. En tant qu'objet de premier niveau du type d'objet `PAGE`, `myframeset` admet la propriété `frameSet` du type d'objet `FRAMESET`. Il crée la balise `<frameset>` et les cadres respectifs ou les jeux de cadres, via les objets de la liste numérique. Pour plus de clarté, leurs attributs ont été standardisés en constantes et déplacés vers le champ **Constants**.

Constants

```
frameSetParams = border="1" frameborder="1" framespacing="0" frameParams = scrolli
ng="auto" frameborder="1" border="1" framespacing="
0" marginheight="0" marginwidth="0" noresize
```

Setup

Le nombre de lignes et de colonnes du cadre ainsi que sa taille sont spécifiés dans les propriétés `rows` et `cols` ; une constante est assignée à `params`.

```
...
myframeset.frameSet.rows = 205,*,50
myframeset.frameSet.params = $frameSetParams
```

Les cadres et jeux de cadres sont à présent définis via la liste numérique. L'objet **10** est du type d'objet **FRAMESET** et sa propriété **obj** est un pointeur vers le TLO représentant, la page **top**. Les attributs du cadre contenus dans la propriété **params** sont aussi des constantes dans notre cas.

```
myframeset.frameSet {
  10 = FRAMESET
  10 {
    obj = top
    params = {$frameParams}
  }
}
```

L'objet **20** du type d'objet **FRAMESET** représente un jeu de cadres imbriqué dans le jeu de cadres principal. Les objets **24** et **26** servent à créer deux cadres et à afficher les pages **menu** et **content** dans le jeu de cadres du milieu.

```
20 = FRAMESET
20.cols = 195,*
20.params = {$frameSetParams}
20 {
  24 = FRAME
  24 {
    obj = menu
    params = {$frameParams}
  }
  26 = FRAME
  26 {
    obj = content
    params = {$frameParams}
  }
}
```

Pour finir, nous créons le cadre **30** qui comprend le TLO **bottom**.

```
30 = FRAME
30 {
  obj = bottom
  params = {$frameParams}
}
```

Si vous appelez à nouveau la page dans votre navigateur sans le paramètre, le résultat correspondra à celui de l'exemple HTML.

5.12.2 Le site exemple avec des cadres

Pour créer le site exemple avec un jeu de cadres, nous avons choisi de diviser la page en trois cadres verticaux dans le gabarit **TS ts wrap template (frames)**. Le menu de gauche, le contenu effectif et les colonnes de droite sont intégrés dans la zone de contenu médiane **content**. Cette disposition facilite les références aux cadres cibles avec les menus. Sur la gauche, ils possèdent tous l'attribut **target="content"**. Le résultat est affiché à la figure suivante. **border** a été fixé à **1** afin de mettre les cadres en évidence.



Figure 5.97:
Site exemple avec des
cadres

Pour ne pas devoir redéfinir à chaque fois les mêmes paramètres des cadres, ils ont à nouveau été saués dans des constantes et définis dans le champ **Constants**.

Constants

```
frameSetParams = border="1" frameborder="1" framespacing="0"
frameParams = scrolling="auto" frameborder="1" border="1" framespacing="0" marginh
eight="0" marginwidth="0" noresize
```

Dans le champ **Setup**, quatre TLO du type **PAGE** sont créés ; chacun d'eux reçoit son propre **typeNum** dans **typeNum**.

Setup

```
myframeset = PAGE
top = PAGE
content = PAGE
bottom = PAGE
```

```
myframeset.typeNum = 0
top.typeNum = 1
content.typeNum = 2
bottom.typeNum = 3
```

Le fichier CSS responsable de la mise en forme est assigné à la propriété **stylesheet** des objets de page censés afficher du contenu.

```
top.stylesheet = fileadmin/styles/ts-template-wrap.css
content.stylesheet = fileadmin/styles/ts-template-wrap.css
bottom.stylesheet = fileadmin/styles/ts-template-wrap.css
```

Le jeu de cadres est appelé via le TLO **myframeset** par la propriété **typeNum=0**. Il est à nouveau défini avec l'objet **frameSet** et ses propriétés. Le nombre de cadres et leur taille sont définis dans **rows** ; **params** prend en argument les attributs des cadres sous forme de constantes du champ **Constants**. Les cadres sont eux-mêmes générés en tant qu'objets de la liste numérique.

```

myframeset.frameSet.rows = 205,*,50
myframeset.frameSet.params = {$frameSetParams}
myframeset.frameSet {
  10 = FRAME
  10 {
    obj = top
    params = {$frameParams}
  }
  20 = FRAME
  20 {
    obj = content
    params = {$frameParams}
  }
  30 = FRAME
  30 {
    obj = bottom
    params = {$frameParams}
  }
}

```

Le jeu de cadres est déjà défini, et seul le contenu des pages doit encore être ajouté. À présent, nous assignons aux pages **top**, **content** et **bottom** des objets temporaires, devant encore être créés.

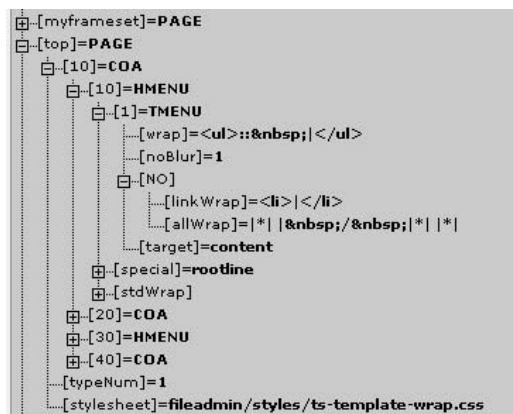
```

top.10 < temp.top
content.10 < temp.content
bottom.10 < temp.bottom

```

Pour définir le contenu, nous profitons du fait que nous pouvons réutiliser des gabarits TS existants. Les menus et les fonctionnalités inclus dans la liste suivante doivent, bien sûr, être incorporés dans l'enregistrement de gabarit avec **Include basis template**. Le gabarit de base **content (default)** est, quant à lui, intégré de la même manière.

Figure 5.98:
TypeScript Object
Browser



Chaque objet temporaire est ici un COA. Ceux-ci doivent être placés au début du champ **Setup** afin d'être définis avant l'analyse syntaxique des objets de page.

Les objets TS désirés sont copiés dans les objets de la liste numérique et enveloppés par la balise `<div>`. Les menus sont ajustés dans le même temps. Par exemple, `10.1.target` reçoit la page `content` en argument. Si vous n'êtes pas sûr des propriétés à utiliser pour les objets emboîtés, le **TypoScript Object Browser** vous aidera.

`temp.top` reprend les éléments de contenu du cadre `top`. Le menu rootline (10), le bouton pour la version imprimable (40), l'image de l'en-tête (20) et la navigation principale (30) sont affichés.

```
temp.top = COA
temp.top {
    ## Menu Rootline
    10 < temp.rootline_autoparser_tswrap
        ## ajustement du menu
    10.1.target = content
    10.stdWrap.wrap = <div id="rootline">|</div>
    ## En-tête
    20 < temp.header_tswrap_autoparser
    20.stdWrap.wrap = <div id="header">|</div>
    ## Navigation principale
    30 < temp.navigation_autoparser_tswrap
        ## ajustement du menu
    30.1.target = content
    30.stdWrap.wrap = <div id="navi">|</div>
    # Version impression
    40 < temp.printversion
    40.stdWrap.wrap = <div id="printversion">|</div>
}
```

Les objets des éléments de contenu de la navigation secondaire (100), l'information méta (200), la colonne `Normal` (20) et la colonne `Droite` (30) sont copiés dans l'objet `temp.content` pour le cadre `content`. Remarquez qu'à l'intérieur du COA `temp.content`, l'objet 10 de la liste numérique a été défini lui-même comme un COA.

```
temp.content = COA
temp.content {
    # Gauche / navigation secondaire + meta
    10 = COA
    10 {
        100 < temp.subnavigation_autoparser_tswrap
            ## Ajustement du menu
        100.1.target = content
        100.2.target = content
        200 < temp.metas_autoparser_tswrap
        200.1.target = content
    }
    10.stdWrap.wrap = <div id="subnavigation">|</div>
    # Contenu gauche
    20 < styles.content.get
    20.stdWrap.wrap = <div id="content">|</div>
    # Contenu droit
    30 < styles.content.getRight
    30.stdWrap.wrap = <div id="right"><div id="rightcontent">|</div><div id="rightfo
```

```

    30.stdWrap.required = 1
  }

```

L'objet temporaire `temp.bottom` pour le cadre `bottom` reprend les propriétés de l'objet `temp`. `copyright_tswrap_autoparser` via l'objet 20, afin d'afficher le sigle copyright dans le bas de page de l'application.

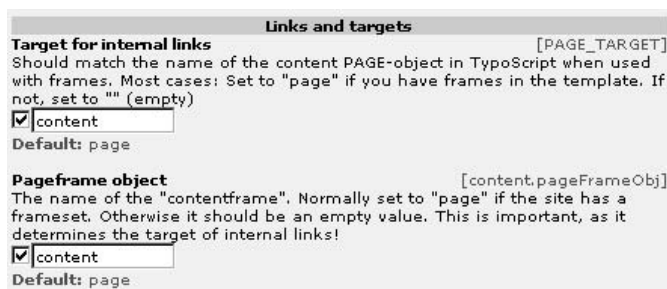
```

temp.bottom = COA
temp.bottom {
  # Bas de page
  10 = TEXT
  10.value = <div id="footer"></div>
  ## Copyright
  20 < temp.copyright_tswrap_autoparser
  20.wrap = <div id="copyright">|</div>
}

```

Pour finir, les cadres cibles sont définis dans le champ Constants pour que les formulaires et les liens des éléments de contenu pointent vers le bon cadre. Ils sont insérés en tant que constantes via `content (default)` avec la valeur par défaut `page`, et vous pouvez les éditer avec le Constant Editor.

Figure 5.99:
Les attributs des liens
sont fixés à content



Constants

```

PAGE_TARGET = content
content.pageFrameObj = content

```

Cet exemple d'utilisation de cadres devrait être suffisant pour s'en faire une première idée. Si vous regardez le résultat dans le frontend, vous verrez qu'il est identique à celui obtenu dans les exemples précédents. Quelques fonctionnalités doivent encore toutefois être ajustées. Par exemple, le menu rootline n'affiche pas encore le chemin de la page courante, mais cela pourrait constituer l'une de vos premières tâches.

5.13 Futur et perspectives

5.13.1 XHTML et accessibilité

Le besoin de séparer le contenu de la forme n'a pas contribué uniquement à la création de systèmes de gestion de contenu, mais a aussi influencé les technologies sous-jacentes. L'HTML a été développé à l'origine en tant que langage basé sur des balises,¹⁰ avec des éléments de mise en page destinés à structurer les documents (les découper en différentes parties), afin d'ajouter d'autres médias tels que des images dans le corps de texte, et afin d'interconnecter les documents par des liens.

La prolifération rapide des sites Web, surtout dans le secteur commercial, a eu une influence sur l'importance du graphisme comme outil de différenciation ou de personnalisation des sites. Pendant longtemps, les tableaux ont été utilisés comme outil principal pour mettre en forme des listes. De plus en plus d'astuces comme celle-ci ont servi à construire les mises en page avec HTML, afin de répondre aux exigences d'un média visuel.

Beaucoup d'astuces et de solutions de rechange, ainsi que les particularités introduites dans les différents navigateurs, ont mis en danger la cohérence de la syntaxe HTML et, par conséquent, l'aptitude des programmes à les interpréter logiquement, tels que les navigateurs destinés aux moins valides. Le consortium WWW dirigé par le développeur HTML Tim Berners-Lee a tenté quelquefois de satisfaire aux exigences du graphisme en étendant les spécifications HTML et en les rendant plus cohérentes, tout en standardisant la syntaxe et en accordant moins de liberté aux navigateurs pour l'interprétation.

L'idée de base derrière HTML, et même derrière son prédécesseur SGML, était précisément de séparer la forme du contenu¹¹. Développées relativement tôt, les feuilles de style en cascade (CSS) constituent une technologie qui respecte cette idée. Mais en pratique, leur acceptation par les éditeurs de navigateurs et les développeurs de sites Web a été très lente.

L'évolution des idées de ces dernières années a été accélérée par les faits suivants :

- La maintenance du contenu et les changements dans le graphisme demandent beaucoup de travail, car les deux sont interdépendants.
- Puisque les documents sont inutilement longs, ils consomment de la bande passante et du temps de chargement.
- La restitution sur d'autres matériels est impossible, si ce n'est à des coûts prohibitifs.
- De nouvelles technologies bien plus flexibles et efficaces telles que le XML ont été développées.
- Des lois sont passées dans plusieurs pays, obligeant à séparer la forme et du contenu pour des raisons d'accessibilité, du moins pour les sites Web des institutions publiques et des autorités.

¹⁰HTML est un dérivé du SGML, qui a été développé comme un langage purement basé sur des balises. Le XML a les mêmes origines, et des efforts sont entrepris pour rétablir XHTML comme un langage de balisage pur.

¹¹« On associe généralement le début du mouvement pour un langage générique à une présentation faite par William Tunnicliffe, président de comité Graphic Communications Association (GCA), à une réunion du Canadian Government Printing Office en septembre 1967 : son sujet — la séparation du contenu des documents de leur forme » dans Charles F. Goldfarb, « The Roots of SGML, a personal recollection », <http://www.sgmlsource.com/history/roots.htm>.

Le XHTML joue un rôle important en tant que standard et comme successeur de l'HTML. Il fait le lien entre l'HTML et le XML. Avec l'introduction du XHTML, les éléments de graphisme ont été, dans une grande mesure, supprimés du langage. La mise en forme devrait être contrôlée exclusivement par CSS, autant que cela est possible, pour séparer plus strictement le contenu de la forme. En outre, la tolérance des fautes est aussi fortement réduite. La syntaxe étant plus stricte, les navigateurs utilisent moins de ressources pour la correction des erreurs et fonctionnent donc de façon très efficace sur beaucoup de dispositifs différents, en ce compris sur les téléphones intelligents.

Les avantages du respect des standards en général, et de XHTML en particulier, sont les suivants :

- Faibles coûts : une meilleure performance due à de plus courts temps de chargement et à une charge plus faible sur le serveur ; simplification de la maintenance des sites Web due à du code plus léger et à la séparation du contenu de la forme.
- Rentabilité de l'investissement : le respect standard assure la compatibilité des nouveaux dispositifs d'affichage et de l'extension à d'autres langages basés sur du XML.
- Plus grand champ d'application : l'accessibilité est améliorée pour différentes plates-formes, navigateurs et technologies d'assistance ; enfin, le dernier avantage, qui n'est pas des moindres, est l'amélioration du positionnement dans les moteurs de recherche.

Référence 895351

Le dernier point explique pourquoi XHTML et l'accessibilité sont souvent mentionnés ensemble : le *Web Content Accessibility Guidelines* (WCAG) ou le *Barrierefreie Informations-technologie-Verordnung* (BITV) allemand recommande le respect des standards, car cela permet un support plus large pour les technologies d'assistance, comme les lecteurs d'écran, les navigateurs vocaux et les lecteurs de lignes en braille pour les personnes ayant des handicaps physiques. Ainsi, en respectant les standards (X)HTML, nous avons déjà fait la moitié du chemin vers un site Web au contenu accessible.

Dans la version 3.6.0., TYPO3 est en conformité avec « XHTML1.0 Transitional ». Tout le code source du cœur a été réaménagé pour respecter ce nouveau standard. Les extensions n'ont pas été impliquées dans ce processus et seront mises à jour progressivement par leurs auteurs respectifs.

Voici les changements essentiels de XHTML1.0 :

- Syntaxe : les éléments ne peuvent pas se chevaucher et doivent avoir des balises fermantes.
- Les noms des éléments et des attributs sont écrits en minuscules.
- Les valeurs des attributs sont toujours placées entre des guillemets anglais, par exemple `border="0"`. Les attributs peuvent ne pas apparaître sous une forme minimale, mais doivent être transcrits en entier, par exemple `checked="checked"` au lieu de `checked`.
- Les éléments vides doivent être délimités, par exemple par `<br />` ou `<br></br>`.
- Le contenu des éléments de script et de style suit une syntaxe précise ou, mieux, est déplacé dans un script externe ou dans des documents de style.

Vérifiez si vos pages Web sont conformes au XHTML avec la validation (X)HTML du consortium WWW (voir la référence ci-contre). Les techniques de génération de pages Web accessibles décrites dans la section suivante fonctionnent en respectant le XHTML, et devraient être considérées dans ce contexte.

Référence 570674

Voir les définitions TypeScript les plus importantes concernant le XHTML configurées via `config` :

Référence 728651

`doctype`

Type de donnée : chaîne de caractères

Exemple : `xhtml_trans` | `xhtml_frames` | `xhtml_strict` | `xhtml_11` | `xhtml_2` | `none`

Déclaration du type de document. Montre les standards que le navigateur devrait respecter.

`doctypeSwitch`

Type de donnée : booléen/chaîne de caractères

Exemple : `1`

Utilisé pour placer le prologue XML en dessous de la déclaration Doctype – une solution de rechange pour certains navigateurs.

`xmlprologue`

Type de donnée : chaîne de caractères

Exemple : `none`

Sert à supprimer le prologue `<?xml version="1.0" encoding="utf-8"?>`.

`htmlTag_setParams`

Type de donnée : `string`

Configure les attributs de la balise `<html>` ; lorsque `doctype` est inséré, tout est déjà correctement défini. Cette propriété peut être utilisée pour éviter certaines incohérences.

`htmlTag_langKey`

Type de donnée : chaîne de caractères

Exemple : `en-US`

Permet de fixer la valeur du langage pour les attributs `xml:lang` et `lang` de la balise `<html>`, à condition que `config.doctype` ait été spécifié.

`htmlTag_dir`

Type de donnée : chaîne de caractères

Exemple : `rtl` | `ltr`

La direction du flux de texte, par exemple pour l'arabe ou l'hébreu.

`removeDefaultJS`

Type de donnée : booléen/string

Exemple : `external` | `1`

JavaScript est soit déplacé ailleurs, soit entièrement supprimé.

`inlineStyle2TempFile`

Type de donnée : booléen

Exemple : `1`

Les définitions de style, de type *inline*, sont déplacées dans des fichiers séparés.

xhtml_cleaning

Type de donnée : chaîne de caractères

Exemple : all | cached | output

Solution de rechange, par exemple pour les extensions.

5.13.2 Accessibilité

En Allemagne, par exemple, le BITV (Barrierefreie Informationstechnik-Verordnung Ordinance on Barrier-Free Information Technology) a été créé suite à la loi sur l'égalité des chances pour les personnes handicapées. Le BITV ne concerne à l'origine que les organismes publics. Sur le plan fédéral, cette ordonnance a été mise en pratique à une large échelle. À la fin de 2005, toutes les institutions fédérales et de nombreuses institutions d'État devront avoir des sites Web accessibles aux personnes handicapées.

Le but premier est de garantir à ces personnes les mêmes chances d'accès à l'information. Même celles ayant un handicap mineur, comme les myopes ou les daltoniens, devraient en bénéficier. Un autre effet positif est l'amélioration générale de l'ergonomie des sites Web, ce qui contribue au bien commun.

Voici un résumé de ces principes directeurs, en accord avec les recommandations WCAG2.0 d'accessibilité les plus récentes du Consortium W3C (qui n'ont pas encore été publiées officiellement).

Pour les personnes handicapées, les sites Web accessibles doivent être :

- Perceptibles (du contenu de rechange comme les attributs alt pour les images)
- Utilisables (avec toute une série de dispositifs et de restrictions physiques, navigation aisée, aide disponible)
- Compréhensibles (langue, abréviations)
- Robustes (conformes aux standards)

En pratique, ces principes directeurs concernent tous ceux qui contribuent à la création d'un site Web basé sur TYPO3 : développeurs du code source, développeurs d'extensions, développeurs de sites Web et rédacteurs. Une bonne partie du travail a déjà été réalisée dans le code source, mais il en reste beaucoup pour les extensions. Toutefois, les développeurs Web et les rédacteurs sont invités à assumer leur part de responsabilité.

L'adaptation pratique la plus importante pour les développeurs du Web est le bannissement dans l'(X)HTML du contrôle de la mise en page. Une des conséquences de cette technique est la gestion de la mise en page sans tableau. La disposition de base, la mise en évidence et les autres techniques de graphisme sont principalement implémentées au niveau du CSS. L'HTML reprend purement et simplement la structure du document. Ceci explique pourquoi le code source de toutes les pages accessibles est tellement semblable.

Par exemple : dans une page HTML conventionnelle, de nombreux développeurs du Web marquent les en-têtes avec des éléments comme `font-size` et `font-weight`. Un lecteur d'écran ne sait que faire avec de tels marquages. Par contre, si les en-têtes sont marqués avec des

éléments **h1-h6**, les lecteurs d'écran les reconnaissent pour ce qu'ils sont, et pourraient par exemple afficher d'abord les en-têtes pour une meilleure navigation dans un grand document.

Outre les propriétés déjà mentionnées ci-dessus, concernant la conformité standard et les déclarations, il existe aussi un certain nombre d'options de contrôle spécifiques au niveau de TypeScript qui touchent à l'accessibilité. En voici quelques exemples :

La configuration suivante désactive la fonction JavaScript dans les menus, afin de supprimer l'affichage d'un cadre peu esthétique autour des images lorsque vous cliquez dessus.

```
noBlur = 1
```

Cette fonction est inoffensive en elle-même. Malheureusement, elle a pour effet d'empêcher la navigation avec la touche de tabulation, et elle doit donc être supprimée.

Le lien partant des étiquettes d'un formulaire et allant vers les éléments adéquats du formulaire doit être structuré de manière claire. Le XHTML fournit un élément `label` pour ce faire. Avec la propriété

```
accessibility = 1
```

du formulaire, nous activons la fonctionnalité qui crée automatiquement des étiquettes et les lie aux éléments de données correspondants. Voici un exemple du résultat :

```
<label for="email">Your Email:</label>
<input type="text" id="email" name="email" />
```

Les contradictions avec le langage prédominant sont précisées dans le code TypeScript :

```
parseFunc.short.Browser = <span xml:lang="en" lang="en">Browser</span>
```

Il en résulte une meilleure lisibilité lorsque vous utilisez des lecteurs d'écran.

À l'intérieur des menus, vous pouvez utiliser

```
accesskey = 1
```

pour créer des attributs, servant à définir les clés d'accès qui permettent de naviguer avec les commandes du clavier. L'utilisation pratique de cet attribut HTML est controversée. En TYPO3, les raccourcis clavier sont créés à partir de la première lettre de l'élément de menu en question. Cela entraîne quelques problèmes, étant donné que de nombreux raccourcis clavier sont déjà réservés par le système d'exploitation et le navigateur. Une opinion largement répandue est que les nombres de 0 à 9 devraient être utilisés comme clés d'accès pour offrir des options de navigation centralisées. Cette fonctionnalité ne peut toutefois être mise en pratique par cette propriété TypeScript : c'est pourquoi vous feriez mieux de ne pas l'utiliser.

Adaptations TYPO3

Les tableaux HTML et les formulaires complexes qui satisfont aux recommandations d'accessibilité ne peuvent pas encore être affichés. Dans de tels cas, les développeurs/administrateurs du

Web devraient stocker statiquement les éléments de contenu préparés à cet effet, par exemple grâce au type de contenu HTML de TYPO3.

Nous pouvons supposer que la conformité standard et l'accessibilité vont jouer un rôle plus important dans la création future de sites Web ; les effets sur TYPO3 sont étudiés par le projet TYPO3 « Accessibility ».

Référence 563588

5.13.3 TemplaVoilà

Référence 003991

Un certain nombre de nouveaux concepts ont été introduits dans TYPO3 par TemplaVoilà¹². Outre la possibilité de préparer des gabarits HTML à partir d'un module backend, vous pouvez aussi créer de nouveaux types de contenus flexibles qui ne sont pas restreints par la structure d'une table de base de données. De plus, le concept de colonne est dépassé ; vous définissez des zones qui n'acceptent que certains types de contenu. Pour ce faire, un nouveau module de page est disponible dans le backend.

Comme nous l'avons déjà mentionné, le développement de TemplaVoilà n'est pas encore terminé, mais cette extension n'en est pas moins un outil puissant qui a déjà fait ses preuves en production, et dont une vue d'ensemble est présentée ici. Une présentation plus détaillée se trouve dans le didacticiel « Futuristic Template Building » disponible à la référence ci-contre.

Même si TemplaVoilà semble convenir parfaitement à l'application, il est en fait constitué de plusieurs composants différents, dont certains sont utilisables séparément. Pour illustrer les nouvelles possibilités, les composants sont présentés individuellement, mais TemplaVoilà peut néanmoins être vu comme un ensemble.

Intégration visuelle des gabarits

Une fonction fondamentale de TemplaVoilà est le traitement et l'intégration aisés de gabarits HTML. Cette fonction est similaire au Template Auto-Parser, excepté que les zones du gabarit à utiliser ou à remplacer sont sélectionnées par un simple clic de souris, grâce à un module backend.

Figure 5.100:
Sélection et
configuration des
zones du gabarit
HTML par TemplaVoilà
avec un clic de souris



¹²Le nom est inspiré d'un jeu de mots « voilà déjà votre gabarit » pour insister sur le fait qu'un gabarit HTML se crée très rapidement dans TYPO3.

Structures de données(DS, Data Structures) et objets gabarits (TO, Template Objects)

Après la mise en correspondance par TemplaVoilà, la configuration du gabarit HTML est sauvée dans deux structures de données XML (DS et TO). Elle contient, à côté de l'information sur le type d'objets ou de données que ces zones acceptent, de l'information sur les zones sélectionnées.

Les structures de données (DS, Data Structures) et les objets gabarits (TO, Template Objects) sont définis séparément mais s'utilisent l'un l'autre. Les structures de données contiennent des définitions abstraites des zones, des champs et des types de champs, comparables à la définition des champs dans une base de données — avec seulement quelques « astuces » supplémentaires.

Les objets gabarits font référence à une DS et définissent l'affichage d'un élément d'une DS. Par exemple, un TO contient de l'information sur les fichiers HTML à utiliser pour l'affichage et, à quel endroit intégrer les valeurs des champs définis dans la structure. Plusieurs TO peuvent être définis pour un DS ; cela permet d'afficher la même information sous différentes formes.

Il y a de plus en plus d'endroits où les DS et les TO sont disponibles. Vous pouvez utiliser les DS pour définir à la fois la disposition de base et les types de contenu. Il est aussi possible d'intégrer TypoScript dans des TO et de fournir des options dynamiques pour les types de contenu qui sont à l'origine plutôt rigides.

La DS suivante (abrégée) définit un champ de sélection de fichiers d'images (TCEforms) et inclut dans le même temps une configuration TypoScript pour leur restitution (<TypoScript>).

```
<T3DataStructure>
  <ROOT type="array">
    <tx_templavoila type="array">
      <title>ROOT</title>
      <description></description>
      <eType>input</eType>
      <tags></tags>
    </tx_templavoila>
    <type>array</type>
    <el type="array">
      <field_image type="array">
        <tx_templavoila type="array">
          <title>Ein Bild</title>
          <eType>image</eType>
          <TypoScript>
10 = IMAGE
10.file.import = uploads/tx_templavoila/
10.file.import.current = 1
10.file.import.listNum = 0
10.file.maxW = 150
10.params = align="right"
          </TypoScript>
        </tx_templavoila>
      <TCEforms type="array">
        <config type="array">
          <type>group</type>
          <internal_type>file</internal_type>
          <allowed>gif,png,jpg,jpeg</allowed>
```

```

        <max_size>1000</max_size>
        <uploadfolder>uploads/tx_templavoila</uploadfolder>
        <show_thumbs>1</show_thumbs>
        <size>1</size>
        <maxitems>1</maxitems>
        <minitems>0</minitems>
    </config>
    <label>Bild</label>
</TCEforms>
</field_image>
</el>
<section>0</section>
</ROOT>
</T3DataStructure>

```

Sélection et restitution de gabarit

La configuration TypoScript la plus simple pour afficher le contenu de la colonne **Normal** ressemble à ceci :

```

page = PAGE
page.typeNum = 0
page.10 < styles.content.get

```

Il manque ici une mise en page de base. Les enregistrements de données de la colonne **Normal** sont affichés au moyen de la configuration par défaut du gabarit standard (par exemple **content (default)**).

TemplaVoilà adopte une approche différente, en redéfinissant complètement la restitution d'une page. La configuration TypoScript correspondante se présente comme suit :

```

page = PAGE
page.typeNum = 0
page.10 = USER
page.10.userFunc = tx_templavoila_pi1->main_page

```

Il n'est pas nécessaire d'inclure des colonnes ici, car la définition des zones de contenu (en colonnes ou non) est définie dans la DS. Elle est sélectionnée dans l'en-tête de la page pour une page ou pour une partie de l'arborescence des pages. Nous choisissons un TO associé à une DS pour déterminer la mise en page.

Le processus de restitution de TemplaVoilà ressemble à ceci :

- Recherche d'une DS et d'un TO dans le rootline.
- Affichage du gabarit.
- Lecture de l'information sur la mise en correspondance des enregistrements de données à partir de l'enregistrement de page courant, pour les zones de la DS qui acceptent les éléments de contenu.
- Restitution des éléments de contenu et affichage de ceux-ci dans les zones adéquates.
- Restitution des éléments de contenu qui contiennent eux-mêmes des zones de contenu.

Définissez dans l'en-tête de page et, pour chaque page, les DS et les TO correspondants à utiliser. Ainsi, vous précisez les zones de contenu et la manière dont ces zones seront restituées.

Assignation du contenu aux pages

De façon surprenante, le contenu incorporé de manière habituelle n'est pas affiché automatiquement avec TemplaVoilà. Dans le module de page de TemplaVoilà, un tel contenu apparaît comme un « enregistrement inutilisé ».

Normalement, associer du contenu à une page se fait simplement en situant les enregistrements dans la page (le champ `pid` de la table de données). Par opposition, TemplaVoilà sauve l'information sur l'affiliation d'un élément de contenu à une page dans l'enregistrement de la page lui-même. Cela signifie que les éléments de contenu de la page peuvent provenir de n'importe quelle autre page et peuvent être réutilisés à de nombreuses reprises, sans devoir créer une copie de ces éléments, et sans devoir intégrer ceux-ci dans la page avec **Insert record**. Pour plus de clarté, le contenu est sauvé dans les pages dans lesquelles il sera publié, comme c'était le cas auparavant.

Flexforms

Les Flexforms vous fournissent un autre moyen pour entrer et sauver des données en TYPO3. Ils sont déjà largement implémentés et sont utilisés de manière extensive par TemplaVoilà.

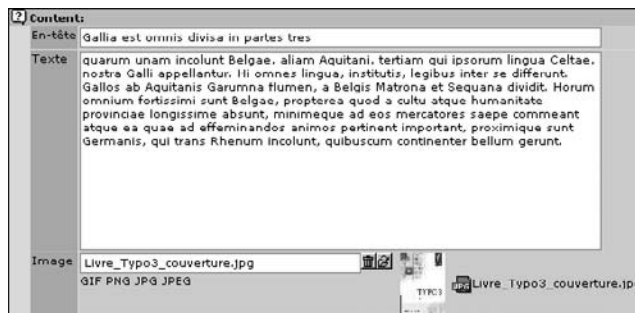


Figure 5.101:
Élément de contenu
avec Flexform et RTE

Si l'on excepte l'utilisation typique de TYPO3 comme CMS, ce système peut être considéré comme un système de gestion de bases de données basé sur une interface Web. En principe, tous les types d'enregistrements peuvent être traités avec TYPO3. Mais on est toujours limité par la structure de la table de base de données correspondante. La table `tt_content` qui sert à sauvegarder plusieurs types de contenu tels que **Texte**, **Image**, **Table** ou **Formulaire** est utilisée de manière flexible — les formulaires pour l'insertion des données de types de contenu sont de taille variable — mais ceci n'est qu'une astuce, puisque chaque enregistrement sauvé contient tous les champs des autres types de contenu, même si ceux-ci ne sont pas utilisés. Si vous voulez sauvegarder des données supplémentaires dans un enregistrement, vous devez étendre la table de base de données.

Les Flexforms apportent une solution grâce à laquelle vous pouvez utiliser pratiquement autant de champs que vous le désirez dans un enregistrement. En outre, chaque enregistrement

peut utiliser d'autres champs. Puisque TYPO3 est construit sur un système de bases de données relationnelles (par exemple MySQL) qui ne peut traiter de telles données, nous avons besoin d'un nouvel artifice. Il consiste à utiliser un champ de base de données suffisamment grand, dans lequel les données sont sauveées dans leur propre format.

Par conséquent, les données de Flexform sont sauveées dans des enregistrements normaux. Dans les cas extrêmes, la table utilisée peut ne contenir qu'un seul champ pour le contenu qui sert à sauver les données. Toutefois, les champs correspondants peuvent servir plusieurs fois et être mélangés à des champs conventionnels.

Les structures de données (DS) déjà familières sont aussi utilisées pour les Flexforms en définissant les types de champs qu'ils utilisent. Les options disponibles dans la structure de données sont les mêmes que celles des TCEForms. Par exemple, le code XML suivant comprend un extrait de la DS pour la définition d'un champ de texte, en intégrant le RTE.

```
<field_newstext type="array">
  <tx_templavoila type="array">
    <title>Newstext</title>
    <eType>text</eType>
    <proc type="array">
      <HSC>1</HSC>
    </proc>
  </tx_templavoila>
  <TCEforms type="array">
    <config type="array">
      <type>text</type>
      <cols>48</cols>
      <rows>5</rows>
    </config>
    <label>Texte</label>
    <defaultExtras>richtext[paste|bold|italic|underline|formatblock|class|left|center|right|orderedlist|unorderedlist|outdent|indent|link|image]:rte_transform[flag=rte_enabled|mode=ts]</defaultExtras>
  </TCEforms>
</field_newstext>
```

Les données du Flexform sont elles-mêmes sauveées sous le format XML. Comme vous le voyez, les composants de base du traitement XML existent déjà dans TYPO3.

On peut se demander si les Flexforms suffisent à eux seuls pour traiter les données. La réponse est qu'il n'est pas facile de sélectionner ou de parcourir les données de la base de données. La base de données ne peut ni traiter les données XML enregistrées, ni distinguer le contenu du code XML qui l'entoure. Dès lors, les Flexforms ne sont pas réellement adéquats pour les données qui doivent être sélectionnées sur base de leur contenu.

Les Flexforms sont aussi disponibles indépendamment de TemplaVoilà.

Contenu flexible

L'intégration visuelle des gabarits HTML, l'utilisation de la DS, les possibilités des Flexforms permettent à TemplaVoilà d'introduire de nouveaux types de contenus flexibles qui viennent s'ajouter aux types de contenu classiques tels que **Text**, **Text w/image**, etc. Ici, le développeur

peut définir assez facilement un nouveau type de contenu à partir d'un gabarit HTML, disponible immédiatement pour le traitement et affiché correctement dans le frontend. De plus, il est possible de définir des zones qui sont reprises à plusieurs endroits. Par exemple, un élément de contenu peut contenir jusqu'à trois liens avec des descriptions.

Vous pouvez aussi utiliser des types de contenu flexibles sans un gabarit d'affichage, même sans TemplaVoilà — dans ce cas, seuls les Flexforms sont utilisés. Nous n'avons besoin que d'une structure de données (DS) pour ce faire, un objet gabarit (TO) n'est donc pas indispensable. C'est une bonne idée que de créer un gabarit simple, et ce, même si ce gabarit sera inutile par la suite, car TemplaVoilà dépend d'un gabarit HTML pour créer des structures de données.

Contenu restrictif

Bien que le nouveau type de contenu se nomme **Flexible content**, vous pouvez utiliser le même concept pour des types de contenu très restrictifs. Avec le nouveau module **Web** → **Page** de TemplaVoilà, une zone de contenu du gabarit est définie de façon à restreindre le nombre de types de contenu spécifiques à deux. Si ces deux types de contenu sont définis spécifiquement pour cette zone, vous avez une solution qui va réagir à l'insertion de contenu non conforme, et ainsi garantir une mise en page universelle à travers tout le site Web. Il s'agit d'un concept opposé à celui que TYPO3 a utilisé jusqu'à présent, dans lequel les rédacteurs jouissaient d'une grande liberté. Vous pouvez bien sûr combiner les deux méthodes de n'importe quelle manière.

Des zones de contenu à la place des colonnes

Même si le concept des colonnes semble toujours pratique et satisfaisant, avec TemplaVoilà, vous n'êtes plus obligé de vous limiter à ce genre de division. Même sans TemplaVoilà, vous pouvez certainement éditer et inclure du contenu dans des zones précises, puisque vous pouvez ajouter et utiliser autant de colonnes que vous le désirez. Cependant, cela devient plus difficile pour les rédacteurs de garder une vue d'ensemble à partir du moment où la mise en page se base de moins en moins sur les colonnes.

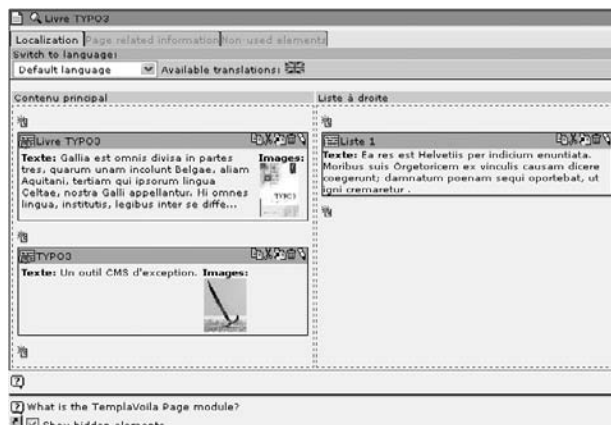


Figure 5.102:
Édition de contenu
avec le module de la
page

Pour cette raison, TemplaVoilà contient un nouveau module de page qui, dans la dernière étape de son développement, est prévu pour afficher schématiquement la mise en page, pour que les différentes zones puissent être facilement éditées. En outre, vous avez la possibilité de configurer des règles pour contrôler l'utilisation de certains types de contenu dans certaines zones.

Utilisation de TemplaVoilà

Référence 673845

Si vous regardez ces possibilités comme un tout, vous voyez que TemplaVoilà convient particulièrement bien aux sites Web qui changent fréquemment de mise en page, pour lesquels les pages sont divisées en zones et en colonnes, ou encore qui utilisent beaucoup de types différents de contenu. En fait, TemplaVoilà a précisément été développé dans le contexte d'un projet ayant ces exigences. Le projet requérait un workflow dans lequel un graphiste Web devait produire des éléments de contenu dans une interface WYSIWYG, à partir de gabarits HTML. Une étude de cas a été réalisée pour ce projet. Elle est disponible à la référence ci-contre.

Comme vous le voyez à la figure 5.103, nous n'avons pas utilisé de colonnes dans le site Web. Nous avons par contre eu très souvent recours à divers gabarits dans le site Web, ayant chacun des zones de contenu différentes.

Figure 5.103:
TemplaVoilà utilisé en
production



TemplaVoilà est un outil puissant pour la mise en forme du contenu et des gabarits et, pour une version alpha, il a atteint un bon niveau de qualité. Il sera certainement encore amélioré pour inclure les fonctions manquantes et en ajouter de nouvelles. À ce stade, TemplaVoilà a déjà introduit plusieurs innovations de base qui peuvent vous servir, même sans utiliser le module.

6 Chapitre

Extensions

6.1 Aperçu

Une des caractéristiques principales de TYPO3 est de pouvoir étendre ses fonctionnalités grâce au système des *extensions*. Les extensions se présentent sous forme de paquets simples à installer pouvant contenir des modules, des plugins, du code TypoScript, etc. Les extensions sont installées sur le serveur, grâce au *gestionnaire d'extensions*, à partir d'un répertoire central, le *répertoire d'extensions*.

Les extensions ont été introduites dans la version 3.5 de TYPO3. Avant cette version, il était déjà possible de compléter le système avec des interfaces. Les extensions dont les tables de base de données débutent avec `tt_` sont issues de cette période. Si ces extensions avaient le mérite d'exister, leur installation était une tâche ardue, impliquant différentes configurations dans plusieurs systèmes de fichiers séparés. Il y avait aussi un risque d'incompatibilité entre les différentes extensions.

L'introduction du système d'extensions a provoqué la création d'une interface d'installation claire. De plus, les interfaces existantes ont été groupées, permettant pour la première fois un

développement décentralisé de TYPO3, lui donnant un second souffle. Depuis l'apparition d'une architecture d'extensions et du répertoire d'extensions, la quantité d'extensions a rapidement augmenté.

Les extensions, disponibles gratuitement, sont développées soit par des programmeurs amateurs ambitieux, soit par des fournisseurs professionnels de services Internet. On y retrouve, en partie, de petites améliorations des fonctions existantes, mais dans la plupart des cas, ces extensions sont des applications à part entière, telles que des archives de publication, des gestionnaires de bibliothèques, des calendriers d'événements, ou un système de réservation pour les hôtels.

Le système d'extensions n'a pas seulement le mérite de faciliter le déploiement d'extensions à l'aide d'une interface d'administration. Il offre aussi aux programmeurs une sécurité et une architecture claire, ce qui garantit la possibilité de mettre à jour le code source de TYPO3.

6.2 Le système d'extensions

Le système d'extensions consiste en plusieurs composants interdépendants :

Extension API

Interface vers le noyau du système, permettant l'intégration d'extensions dans le système TYPO3

Gestionnaire d'extensions

Module backend pour l'administration et l'installation d'extensions

Répertoire d'extensions

Répertoire central en ligne permettant le téléchargement d'extensions à partir de, ou vers le site TYPO3.org

Ces composants forment la base des extensions – mais à quoi ressemblent-ils concrètement ?

6.2.1 Structure d'extensions

Une extension consiste en plusieurs fichiers rassemblés dans un répertoire. Le nom du répertoire représente aussi la *clé d'extension* pour cette extension. Les sous-répertoires sont utilisés pour les composants (plugins, modules, etc.) de cette extension. Voici le répertoire pour l'extension mininews :

```
mininews /
  doc /
    manual.sxw
    wizard_form.dat
    wizard_form.html
  ext_emconf.php
  ext_icon.gif
  ext_localconf.php
  ext_tables.php
  ext_tables.sql
  ext_typoscript_setup.txt
  icon_tx_mininews_news.gif
```

```

locallang.php
locallang_db.php
pil/
    ce_wiz.gif
    class.tx_mininews_pil.php
    class.tx_mininews_pil_wizicon.php
    clear.gif
    locallang.php
tca.php

```

Seul le programmeur manipule ces fichiers individuellement, s'il désire apporter des modifications directement dans l'extension. Un administrateur ou programmeur de site Web n'accède même pas à ces fichiers, ou n'a pas à s'en inquiéter, puisque les extensions sont installées en paquetages, comme nous allons le voir.

6.2.2 Clé d'extension

Le nom du répertoire d'une extension définit sa clé d'extension, et cette clé constitue la base des noms des fichiers et du code PHP à l'intérieur de l'extension. Ceci signifie que si vous renommez le répertoire (ou la clé), vous devez aussi renommer les fichiers, et modifier le code PHP.

La clé d'extension doit être unique car toutes les extensions sont installées dans un seul répertoire. Dans un même temps, un espace de nommage au sein de TYPO3 est défini par les clés uniques, permettant d'éviter les conflits. Les clés d'extension doivent être enregistrées dans le répertoire d'extensions de TYPO3 (TER). Nous y reviendrons plus loin.

6.2.3 Composants d'extensions

Pour augmenter les capacités du système, une extension peut contenir les composants suivants :

- Nouvelles tables de base de données* ¹.
- Extension de tables de base de données existantes*
- Tables de base de données avec des données statiques
- Fonctions frontend
- TypoScript*
- Éléments de contenu*
- Menus frontend*
- TypoTags*
- Plugins frontend de toutes sortes*
- Fonctions backend
- Modules backend*

¹ Les composants marqués d'une astérisque (*) peuvent être créés à l'aide de l'*Extension Kickstarter*, qui simplifie considérablement le développement de telles extensions. Le Kickstarter est décrit en détail à la section 7.2.

- Entrées dans des menus contextuels*
- TSConfig page et utilisateur*
- Services*
- Bibliothèques
- Configuration système
- Habillage backend (couleurs et icônes)
- Extension/modification de chaque classe PHP du système

6.2.4 Catégories d'extensions

Les extensions sont classées en catégories de base, en fonction de leur place dans l'architecture TYPO3. Ces catégories sont affichées dans le gestionnaire d'extensions et correspondent fondamentalement à une division technique. Elles sont aussi classées dans le répertoire d'extensions en fonction de leur application. On y retrouve des catégories telles que *communication*, *eCommerce*, *habillage*, *administration*, et autres.

Backend

Les extensions listées dans cette catégorie complètent la fonctionnalité du backend, mais n'apparaissent pas en tant que modules.

Backend Modules

Les modules backend y sont listés ; on y retrouve de nouveaux modules principaux (ex. : **Web**, **Outils**), des modules (ex. : **Web** → **Liste**) ou des fonctions de sous-modules (ex. : **Web** → **Fonctions** → **Importer**).

Frontend

Les extensions de cette catégorie contiennent de petites fonctionnalités ou configurations frontend (balises méta, nouvelles balises TYPO3)

Frontend Plugins

Cette catégorie contient toutes les extensions qui complètent l'affichage et les fonctionnalités du frontend comme des éléments de contenu, des menus, des bordures de texte ou des applications frontend complètes (livre d'or, actualités, etc.).

Miscellaneous

Vous y trouverez toutes les extensions qui n'entrent pas dans d'autres catégories, telles que des tables de base de données statiques, des fonctions pour créer des fichiers PDF ou programmer des bibliothèques.

Services

Cette catégorie contient des fonctions pouvant être utilisées par d'autres extensions ou par le système. Ces fonctions représentent une autre interface d'extension dont la complexité est moindre que celle des extensions.

Templates

Dans cette catégorie sont regroupés les gabarits TypoScript de sites entiers sous forme d'extensions. Ils correspondent aux gabarits des sites compris dans TYPO3, se trouvant dans les enregistrements de gabarits sous **Static templates**.

Exemples

Catégorie pour exemples de tous types ; ce sont souvent des extensions construites dans des didacticiels ou illustrant l'utilisation d'une API.

Documentation

Cette catégorie contient des extensions de documentation basées sur des documents OpenOffice.

6.2.5 Installation: niveau système, global ou local

Les extensions peuvent être installées dans trois répertoires différents du système :

`typo3/sysex/` (type : *système*)

On y trouve les extensions système telles que `cms` et `lang`. Ici, aucune extension ne peut être installée avec le gestionnaire d'extensions.

`typo3/ext/` (type : *global*)

On y retrouve les extensions globales. Ces dernières sont normalement celles fournies par TYPO3. Ces extensions contiennent surtout des modules backend et fournissent des fonctionnalités de base ou étendues nécessaires à la plupart des applications TYPO3. Il est possible de configurer l'option `allowGlobalInstall` à l'aide de l'**Installation Tool** afin de permettre l'installation d'extensions dans ce répertoire. Les paramètres par défaut ne l'autorisent pas, parce que des conflits entre les versions pourraient apparaître dans un environnement où plusieurs sites TYPO3 partagent une même installation TYPO3.

`typo3conf/ext/` (type : *local*)

Les extensions sont normalement toutes installées dans ce répertoire. On les appelle extensions locales, parce qu'elles ne sont valables que pour l'instance TYPO3 locale. En fait, elles ont même priorité sur les extensions globales. Cela signifie que lorsqu'une extension est installée localement et globalement, l'installation locale est toujours comprise dans le système, même si son numéro de version est inférieur à celui de l'installation globale.

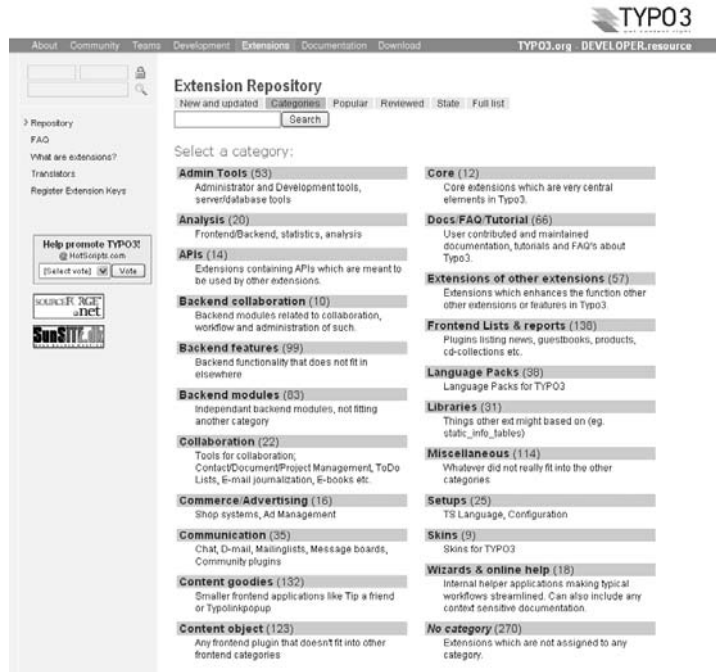
Plusieurs instances (sites Web) TYPO3 peuvent partager une même installation TYPO3 en utilisant des liens symboliques, par le biais desquels ils partagent aussi des extensions globales. Toutefois, les extensions locales ne demeurent visibles que pour une instance particulière d'un site.

6.2.6 Répertoire d'extensions

Dans le *répertoire d'extensions TYPO3* (TER), les extensions sont enregistrées de façon centrale et peuvent être téléchargées vers votre propre installation TYPO3, puis installées. Le répertoire se trouve sur le site de TYPO3.org. Les programmeurs peuvent transférer une extension vers le répertoire en quelques clics de souris via le gestionnaire d'extensions. L'extension est alors disponible soit pour certains utilisateurs, soit pour tous, en fonction de sa configuration. Il est tout aussi simple de télécharger une extension, en quelques clics seulement.

Référence 617220

Figure 6.1:
Aperçu des catégories
d'extensions dans le
TER sur TYPO3.org



Lorsqu'une extension a été téléchargée et installée, la liste des extensions disponibles dans le répertoire d'extensions de TYPO3.org est reprise dans le module **Gest d'Extensions** de votre propre installation TYPO3. Lorsqu'une extension est sélectionnée et installée, le répertoire envoie un fichier (d'extension .t3x), contenant le répertoire complet de l'extension, avec tous les fichiers nécessaires, vers le gestionnaire d'extensions de l'installation locale. Ce dernier désarchive le paquetage vers le répertoire d'extensions. L'extension est maintenant prête à être installée à l'aide du gestionnaire d'extensions. Lors de l'installation, certains fichiers du répertoire d'extensions sont détectés, lus et insérés dans le système. Le gestionnaire d'extensions fournit aussi des fonctions d'installation pour créer les tables de base de données nécessaires, ou offrir une sélection d'options de configuration à l'utilisateur.

La voie inverse, c'est-à-dire l'envoi d'une extension vers le répertoire, fonctionne d'une façon similaire et est expliquée à la section 7.3.3.

Le processus d'installation est très simple et pratique pour l'utilisateur : si une extension est installée, l'utilisateur ou l'utilisatrice trouvera de l'information dans les fichiers installés, mais ce n'est pas nécessaire. Vous pouvez vérifier qu'une extension est déjà installée grâce à l'icône verte dans le gestionnaire d'extensions en mode **Loaded Extensions** ou **Available extensions to install**.

Au moment de la publication de ce manuel, il existe plus de 1000 extensions disponibles. Toutefois, une grande quantité des extensions enregistrées dans le répertoire ne sont pas disponibles pour le public, probablement parce qu'elles n'ont pas été correctement complétées, ou parce que les programmeurs continuent d'en améliorer le code, ou encore parce qu'au-

cune documentation n'existe. Nous souhaitons toutefois souligner que les projets OpenSource vivent grâce aux contributions actives des utilisateurs, et qu'il est déplorable de penser que certains de ces diamants bruts pourraient, s'ils étaient publiés, retrouver un nouvel éclat dans les mains d'autres programmeurs ! Il est aussi beaucoup plus simple de trouver de l'aide et du soutien si vous possédez déjà une version beta appropriée, ou une description de projet.²

6.2.7 Documentation

Dans le répertoire d'extensions, vous trouverez aussi la documentation complète de TYPO3, sous forme de documents OpenOffice. Ces extensions ne contiennent que de l'information, et appartiennent à la catégorie documentation. Les documents sont disponibles sur TYPO3.org en tant que pages HTML dans la section « Documentation », où vous pouvez aussi ajouter des commentaires.

Référence 280413

En dehors de ces extensions de documentation, qui contiennent généralement des références et des didacticiels, il peut, et devrait, y avoir de la documentation spécifique pour chacune des extensions. Le même système que pour les extensions de documentation s'applique ici, hormis que ces extensions contiennent aussi un plugin, un module ou d'autres données. La documentation spécifique à une extension peut, si nécessaire, être téléchargée à partir du répertoire en même temps que l'extension. On la retrouve ensuite dans le fichier OpenOffice `manual.sxw` dans le sous-répertoire `doc/` de l'extension.

Les documents OpenOffice proposent certains avantages sur d'autres formats tels que HTML, PDF ou DocBook :

- Un traitement de texte puissant et disponible gratuitement facilite la rédaction.
- Les fichiers OpenOffice `.SXW` sont imprimés dans le format désiré.
- Il est possible de créer des fichiers PDF directement à partir d'OpenOffice.
- Le format de fichier OpenOffice qui est ouvert et basé sur XML est simple à lire et à convertir (en HTML par exemple).
- Il est possible d'afficher la documentation directement en format HTML dans TYPO3.org, grâce à *Document Suite*.

6.3 Gestionnaire d'extensions

Le gestionnaire d'extensions (EM³) est un module backend prévu pour l'administration des extensions. Le module est situé dans le module principal **Outils**, ce qui implique qu'il est normalement réservé aux administrateurs. Les fonctionnalités du gestionnaire d'extensions comprennent :

- La liste des extensions installées
- L'installation et la désinstallation des extensions

²Les descriptions de projet se trouvent dans la section **Projects** du site TYPO3.org, au même titre que les projets déjà à la recherche d'aide et de sponsors. N'importe qui peut écrire une description de projet, ce qui constitue une façon efficace d'aborder des idées et des concepts afin de les mettre ensuite en pratique sur TYPO3.org.

³NdT : EM est l'abréviation d'*Extension Manager*

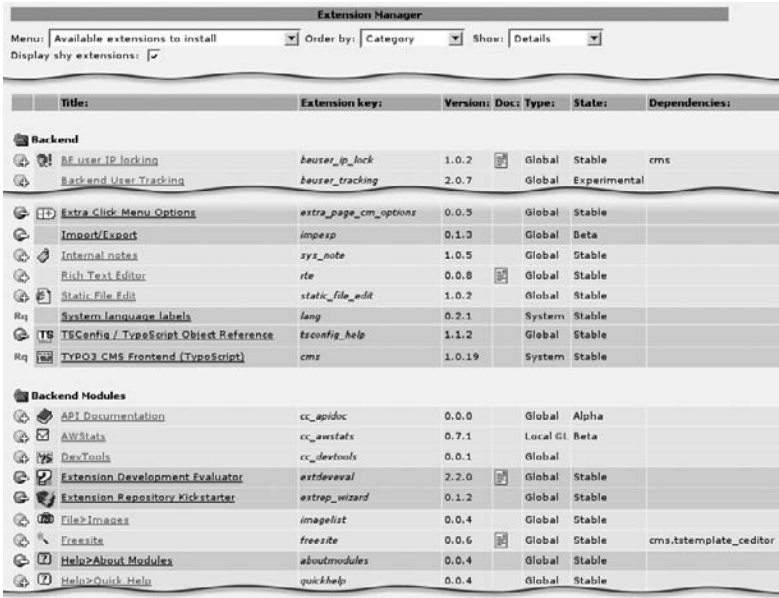
- Le transfert des extensions vers votre propre installation par un fichier T3X
- Le transfert des extensions du répertoire en ligne (TER) vers votre propre installation (téléchargement)
- Le transfert de vos propres extensions vers le répertoire d'extensions TYPO3
- Le développement de vos propres extensions (Kickstarter)

La section suivante présente les fonctions et les écrans correspondants du gestionnaire d'extensions.

6.3.1 Liste des extensions disponibles

Ce mode affiche toutes les extensions disponibles dans l'installation TYPO3. Elles sont listées dans le tableau avec de l'information complémentaire. L'icône +/- située au début de chaque ligne indique si une extension est installée (icône verte). En cliquant sur cette icône, vous installerez ou désinstallerez l'extension correspondante.

Figure 6.2:
Affichage des
extensions
disponibles dans le
gestionnaire
d'extensions



The screenshot shows the 'Extension Manager' window. At the top, there are filters: 'Menu: Available extensions to install', 'Order by: Category', and 'Show: Details'. Below this is a table with columns: Title, Extension key, Version, Docs, Type, State, and Dependencies. The table is divided into sections: 'Backend' (containing 'User ID Locking' and 'Backend User Tracking'), 'Extra Click Menu Options', 'Image/Export', 'Internal notes', 'Rich Text Editor', 'Static File Edit', 'System Language Labels', 'TSConfig / TypeScript Object Reference', and 'TYPO3 CMS Frontend (TypeScript)'. Below these are 'Backend Modules' including 'API Documentation', 'AWS Stats', 'DevTools', 'Extension Development Evaluator', 'Extension Repository Kickstarter', 'ImageList', 'Freemove', 'Hello>About Modules', and 'Hello>Quick Help'.

Title	Extension key	Version	Docs	Type	State	Dependencies
Backend						
⊕  User ID Locking	beuser_id_lock	1.0.2		Global	Stable	cms
⊕  Backend User Tracking	beuser_tracking	2.0.7		Global	Experimental	
Extra Click Menu Options						
⊕  Extra Click Menu Options	extra_page_cm_options	0.0.5		Global	Stable	
Image/Export						
⊕  Image/Export	impexp	0.1.3		Global	Beta	
Internal notes						
⊕  Internal notes	sys_note	1.0.5		Global	Stable	
Rich Text Editor						
⊕  Rich Text Editor	rte	0.0.8		Global	Stable	
Static File Edit						
⊕  Static File Edit	static_file_edit	1.0.2		Global	Stable	
System Language Labels						
⊕  System Language Labels	lang	0.2.1		System	Stable	
TSConfig / TypeScript Object Reference						
⊕  TSConfig / TypeScript Object Reference	tsconfig_help	1.1.2		Global	Stable	
TYPO3 CMS Frontend (TypeScript)						
⊕  TYPO3 CMS Frontend (TypeScript)	cms	1.0.19		System	Stable	
Backend Modules						
⊕  API Documentation	ccapidoc	0.0.0		Global	Alpha	
⊕  AWS Stats	cc_awsstats	0.7.1		Local GL	Beta	
⊕  DevTools	cc_devtools	0.0.1		Global		
⊕  Extension Development Evaluator	extdeveval	2.2.0		Global	Stable	
⊕  Extension Repository Kickstarter	extrop_wizard	0.1.2		Global	Stable	
⊕  ImageList	imagelist	0.0.4		Global	Stable	
⊕  Freemove	freemove	0.0.6		Global	Stable	cms.ttemplate_editor
⊕  Hello>About Modules	aboutmodules	0.0.4		Global	Stable	
⊕  Hello>Quick Help	quickhelp	0.0.4		Global	Stable	

Chaque extension est affichée avec son nom et son icône propres. Quelques-unes ont un point d'interrogation pour icône, ce qui signifie que le programmeur de l'extension n'a pas encore créé d'icône pour celle-ci.

Les autres détails affichés sont la clé d'extension ainsi que le numéro de version. Dans la colonne suivante, une icône document peut apparaître. Elle signifie qu'un fichier OpenOffice **manual.sxw** existe dans le sous-répertoire **doc/** de cette extension. La documentation se trouve aussi sur TYPO3.org dans la liste d'extensions.

Il est possible d'installer une extension à la fois globalement et localement. Une telle installation est marquée comme **Local GL** dans le gestionnaire d'extensions. De cette façon, les

extensions installées localement peuvent être modifiées selon vos propres exigences, et vous pouvez être assuré que cette version-là sera intégrée dans le système, plutôt que l'originale.

6.3.2 Importer des extensions du répertoire

Les extensions peuvent être importées simplement à partir du répertoire en ligne vers votre installation. Pour ce faire, sélectionnez l'option **Import extensions from online repository** du menu principal du gestionnaire d'extensions.

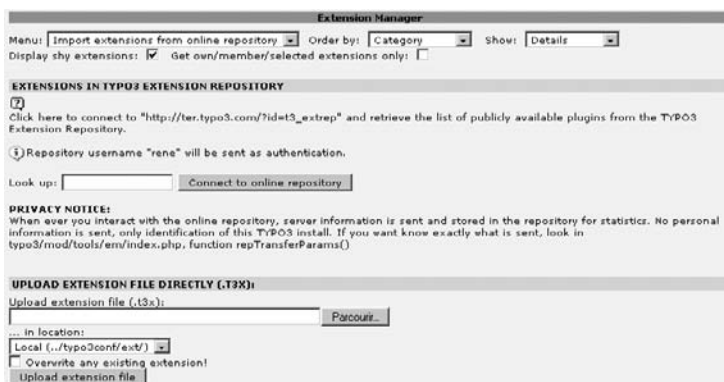


Figure 6.3:
Connexion au
répertoire et transfert
d'un paquetage
d'extension vers le
serveur

Avant d'aller plus loin dans notre description du transfert d'extensions à partir du répertoire, nous expliquerons brièvement comment un fichier de paquetage d'extension (*.t3x) peut être transféré directement vers votre serveur à l'aide du bouton **Upload extension file**. Vous devez sélectionner le fichier T3X ainsi que le répertoire d'installation, puis transférer le fichier vers le serveur en cliquant sur **Upload extension file**. L'installation comme telle correspond à la reprise des données en provenance du répertoire, et est décrite ci-dessous.

Vous pouvez accéder au répertoire d'extensions en cliquant sur **Connect to online repository**. Une connexion est alors établie vers le répertoire et vous obtenez une liste des extensions disponibles. De plus, vous pouvez spécifier une clé dans le champ **Look up**, avec laquelle vous pouvez accéder à des versions spéciales des extensions. Une version d'extension protégée de cette manière n'est pas publique, et n'est accessible qu'en spécifiant la clé. C'est utile si vous voulez communiquer une extension aux testeurs d'une équipe de développement.

La liste en provenance du répertoire d'extensions contient d'autres informations en plus de celles déjà mentionnées. La première colonne affiche des icônes de téléchargement si l'extension n'est pas encore disponible sur le serveur local, ou si une version plus récente est disponible. La colonne du centre vous permet de comparer le numéro des versions. La colonne **Access** montre si vous êtes le propriétaire (**Owner**) de l'extension ou un membre (**Member**). Les membres d'une extension peuvent télécharger celle-ci, même si elle n'est pas disponible dans le répertoire pour tous les utilisateurs. Les différents états (disponible localement ou non-disponible, propriétaire/membre)⁴ sont aussi mis en évidence par une couleur différente des lignes du tableau.

⁴Les fonctions des membres ne sont présentes que si vous avez un compte utilisateur sur TYPO3.org et que vous avez saisi les données d'accès dans le menu **Settings** du gestionnaire d'extensions.

Figure 6.4:
Liste des extensions
disponibles dans le
répertoire
d'extensions

	Title:	Extension key:	Version:	Cur. Ver:	Cur. Type:	Access:	TS vers:	PHP:	Size:	DL:	State:	Dependencies:
Backend												
	Admin Panel Wrapping/Positioning	ingmar_adminpanelwrapping	1.0.3				3.5.0	4.3.2	8.5 K/2.9 K	2950/1462	Stable	
	Alternative RTE	alternativa_rte	0.0.1	0.0.0	Local		3.5.0	4.3.0	716 K/162 K	353/353	Experimental	cms
	BE Autoleap	julie_beloginot1	0.0.4	0.0.4	Local		3.5.0	4.1.2	54 K/26 K	1468/963	Stable	
	BE user IP locking	beuser_ip_lock	1.0.2	1.0.2	Global		3.6.0-dev	4.2.3	28 K/23 K	934/830	Stable	cms
	Bigger Backend Font	bigger_backend_font	0.1.2				3.5.0	4.1.2	22 K/15.2 K	350/327	Beta	cms
Frontend												
	404 Error Page Handling	error_404_handling	0.0.2				3.6.0RC1	4.3.1	57 K/26 K	524/84	Beta	cms
	Alaspin	gst_alaspin	0.4.2				3.5.0	4.2.3	37 K/23 K	899/218	Beta	cms
	Always for Images	imgc_image_always	0.1.3				3.5.0	4.3.0	111 K/92 K	3819/2294	Stable	cms
	Front End User Admin	feuser_admin	1.0.2	1.0.2	Global		3.5b4	4.1.2	23 K/5.6 K	914/878	Stable	cms
	Front End User List	gst_feuser_list	0.6.0				3.5.0	4.2.2	39 K/22 K	216/218	Beta	cms
	Front End User Registration	sr_feuser_register	0.2.0				3.5.0	4.3.3	232 K/61 K	173/124	Beta	cms, sr_static_info
	Frontend Extender	dp_feedit	0.1.8	0.0.4	Global	Member	3.6.0RC1	4.3.3	182 K/48 K	24/2	Alpha	cms
	Frontend Extender Test suite	dp_feedit_test	0.1.8	0.0.2	Global	Member	3.6.0-dev	4.3.2	187 K/20 K	8/2	Stable	dp_feedit
	Frontpage Teaser	fr_teaser	0.0.1				3.5.0	4.1.2	43 K/9.9 K	397/397	Beta	cms
	GFE Forms	gforms	0.0.1				3.5.0	4.2.3	300 K/44 K	899/901	Alpha	cms
	It Gallery	it_gallery	0.5.1				3.5.0	4.2.3	575 K/449 K	6023/4364	Beta	cms, it_table
	General Office Display	rlmp_officeimport	1.0.4	1.0.4	Global		3.6.0-dev	4.1.2	1.0 M/852 K	2254/1538	Stable	cms, libunscipped
	Glossary	oc_glossar	0.2.1	0.2.0	Global	Owner	3.5.0	4.2.3	342 K/98 K	19/7	Alpha	cms, everlib
	Google API Search	google_api_search	0.0.4				3.5.0	4.1.2	47 K/15.5 K	724/652	Experimental	cms, nusoap
	Graphic Visitor Counter	big_fe_graphiccounter	0.1.6	0.1.6	Local		3.5.0	4.2.2	153 K/6.7 K	1751/463	Beta	cms
	Guestbook	it_guest	1.0.8	1.0.5	Global		3.6.0RC1	4.2.3	78 K/43 K	1151/306	Stable	cms

Dans tous les écrans, on peut cocher l'option « Shy extensions » pour afficher les extensions qui composent le paquetage de base, disponibles dans la plupart des installations. Les menus **Order by** et **Show** permettent d'accéder à d'autres modes d'affichage.

Si vous avez décidé de télécharger une extension, celle-ci est d'abord transférée sur votre serveur, mais pas immédiatement installée. Pour l'installer, cliquez sur l'icône + qui apparaît dans la liste, après que l'extension a été téléchargée. Vous avez aussi la possibilité d'installer l'extension dans le mode **Available extensions to install** à l'aide de l'icône +.

Vue détaillée et options

Si vous souhaitez avoir plus de détails sur une extension avant de la télécharger, cliquez sur son nom. Vous verrez alors un aperçu précis donnant des informations telles que son nom, l'auteur, la version, mais aussi les fichiers et les tables de base de données qu'elle contient.

Dans la vue détaillée, vous pouvez sélectionner la version à télécharger. Si l'option **allow-GlobalInstall** est activée dans le \$TYPO3_CONF_VARS (Outil d'installation), vous pouvez aussi sélectionner le chemin pour l'installation des extensions globales dans **typo3/ext/**. Si l'option **em_alwaysGetOOManual** est aussi activée, la documentation de l'extension sera automatiquement téléchargée en même temps, action qui, sinon, devrait être réalisée séparément. L'option **Include most recent translation** permet d'introduire toutes les traductions disponibles dans l'extension. Ce qui peut être utile si le programmeur n'a pas encore intégré les traductions souhaitées, disponibles via le TER, dans la version considérée.

En fonction du type d'extension et des modules et plugins qu'elle contient, une ou plusieurs étapes seront nécessaires à l'installation. Quelques extensions peuvent être installées en un

seul clic. Pour les autres, on vous demande si vous voulez effectuer les étapes d'installation nécessaires, telles que par exemple créer de nouvelles tables de base de données, ou vider le cache. Il est parfois nécessaire de mettre en place la configuration et de la modifier pendant l'installation. Les paramètres pouvant être modifiés sont décrits, soit sur le moment même, soit dans la documentation fournie avec l'extension.



Figure 6.5:
Vue détaillée de
l'extension General
Office Dispatcher

Toutefois, après avoir installé les modules, ces derniers ne sont pas immédiatement visibles dans le backend. Vous devez alors recharger le backend en cliquant sur le bouton « rafraîchir » de votre navigateur (ou en utilisant le raccourci clavier **(Ctrl)+(R)**).

Les extensions sont désinstallées de façon similaire, en cliquant sur l'icône verte « - ». Assurez-vous que les tables de base de données utilisées seulement par cette extension ne sont pas automatiquement effacées. De cette façon, vous pouvez conserver les données. Par contre, à long terme, il est probable que plusieurs tables inutilisées s'accumulent dans la base de données. Elles peuvent alors être enlevées à l'aide de l'*outil d'installation* que nous avons présenté plus tôt.

6.3.3 Le Kickstarter

Le Kickstarter est l'outil pour le développement d'extensions, permettant la mise en place rapide et simple du cadre dans lequel une extension sera créée.

Le Kickstarter est enregistré dans une extension séparée (répertoire d'extension Kickstarter : `extrep_wizard` ?) ; si elle n'est pas disponible dans votre installation, commencez par télécharger l'extension. Dès qu'elle est installée, le Kickstarter apparaît dans le menu **Make new extension** du gestionnaire d'extensions.

Figure 6.6:
Le Kickstarter prêt à
créer une nouvelle
extension

The screenshot shows the 'Extension Manager' window. At the top, there's a 'Menu:' dropdown set to 'Make new extension'. Below this is the 'KICKSTARTER WIZARD' section, which contains a list of steps, each with a plus icon to its right:

- General info
- New Database Tables
- Extend existing Tables
- Frontend Plugins
- Backend Modules
- Integrate in existing Modules
- Clickmenu items
- Services
- Static TypoScript code
- TSconfig
- Setup languages

Below the list, there's a text input field labeled 'Enter extension key:'. Underneath the input field is a note: 'Make sure to enter the right extension key from the beginning here! You can register one here.' At the bottom of the wizard section, there are two buttons: 'Update...' and 'Total form'.

Le chapitre suivant explique comment utiliser le Kickstarter et les fonctions du gestionnaire d'extensions prévues pour les programmeurs.

7

Chapitre

Développement d'extensions

TYPO3 est un framework pour les applications Web. On peut l'étendre, l'adapter, ou même le changer complètement grâce aux extensions. Le noyau de TYPO3 est un système de gestion de contenu Web reposant sur une base de données. On le remarque par exemple au fait que le CMS est mis en place sous forme d'extension, ce qui signifie qu'il est une application de TYPO3. Cette séparation entre noyau et applications réduit la complexité et facilite un développement continu du système, grâce à une API d'extension bien définie et bien structurée. De plus, chaque extension possède son propre espace de nommage. La combinaison de ces deux propriétés évite les conflits et permet d'étendre TYPO3 dans presque toutes les directions.

Le gestionnaire d'extensions a déjà été présenté au chapitre 6. Nous avons alors expliqué les différents types d'extensions existants et la façon de les mettre en ligne.

Dans ce chapitre, nous verrons comment programmer par vous-même différents types d'extensions (plugins, modules, ...). Pour y parvenir, vous avez besoin de bases solides en PHP, en programmation orientée objet, et en SQL. Même si vos connaissances dans ces domaines ne sont pas très étendues, n'ayez pas peur de lire ce chapitre : TYPO3 constitue un bon moyen de

se familiariser avec les langages PHP et SQL. Dans ce cas, nous vous recommandons toutefois d'avoir de la documentation plus précise à portée de main.

Nous tenons à souligner que ce chapitre est une introduction au développement de TYPO3, et qu'il ne fournit qu'un aperçu. Même si le Kickstarter prend à son compte une bonne partie du travail du programmeur, il sera quand même parfois nécessaire de s'attaquer à la documentation. La plupart des sections débutent avec une référence vers d'autres sources d'information.

7.1 Un compteur de visiteurs en 20 minutes

Afin d'illustrer combien il est simple et rapide d'étendre TYPO3, nous allons programmer un plugin simple. Notre but étant seulement de donner un aperçu général, nous ne donnerons pas d'explications détaillées dans cet exemple.

Nous allons programmer un compteur de visiteurs, l'invention qui, à l'âge de pierre du Web, a probablement fait la joie de millions d'auteurs de pages d'accueil. Le plugin utilise une table de base de données pour enregistrer le statut du compteur. Le compteur ne devrait compter chaque visiteur qu'une fois ; on y parvient grâce à la session utilisateur, créée et gérée automatiquement par TYPO3, et ce, même si un visiteur visite plusieurs fois la même page au cours d'une session.

On insère normalement un plugin dans une page à l'aide de l'élément de contenu **Insérer un plugin**, mais nous verrons qu'il existe d'autres façons d'utiliser un plugin.

Le plugin doit être prêt en 20 minutes. Alors allons-y !

00 :00

Nous avons d'abord appelé le backend, et nous sélectionnons maintenant le Kickstarter, auquel on accède par le gestionnaire d'extensions via le mode **Make new extension**.

Il faut d'abord spécifier une clé d'extension. Nous utiliserons `user_visitcounter`. Rafraîchissons maintenant le formulaire en cliquant sur **Update**.

Figure 7.1:
Information générale
sur l'extension sous
General Info

The screenshot shows the 'Extension Manager' window with the 'Kickstarter Wizard' active. The 'General Info' tab is selected. The form contains the following fields and options:

- Title:** Visitor Counter
- Description:** Counts visitors and display a count
- Category:** Frontend Plugin
- State:** Alpha (Very initial development)
- Author Name:** (empty)
- Author email:** (empty)

At the bottom, there are buttons for 'Update...', 'Total form', 'View result', 'D/L as file', and a checkbox for 'Print WOP comments'.

Une extension peut comporter plusieurs éléments : plugins, modules, table de base de données, etc. Débutons en ajoutant l'élément **General info**, qui contient de l'information générale à propos de l'extension. Nous remplissons maintenant le formulaire, et nous enregistrons l'entrée en cliquant sur **Update**. Le compteur a besoin d'une table de base de données pour pouvoir compter ; nous la créons avec l'option **New Database Tables**.

La table a besoin d'un nom explicite, qui sera ensuite affiché dans le backend. Nous l'appellerons **Visitor Counter**. Derrière le champ d'entrée, le mot **[english]** apparaît. Il nous rappelle que nous souhaitons aussi avoir des descriptions dans d'autres langues. Nous ajoutons donc le français à cette extension en cliquant sur **Setup languages** et nous sélectionnons **French**.

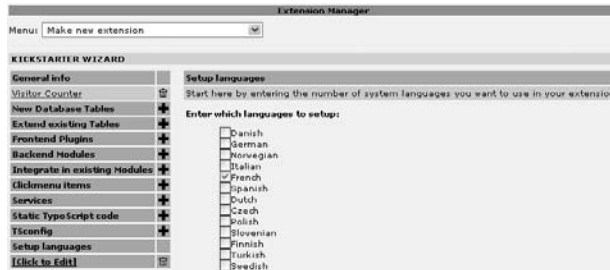


Figure 7.2:
Édition de langues
supplémentaires dans
le Kickstarter

Cependant, la table n'est pas encore terminée, et doit encore être traitée. Nous sélectionnons la table **Visitor Counter** dans le menu **New Database Tables**, qui est déjà listée en tant que composant de cette extension. Nous pouvons maintenant spécifier pour celle-ci un nom français.

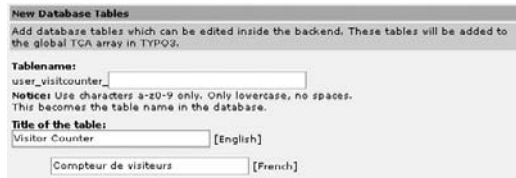


Figure 7.3:
Nouvelle table de
base de données avec
un nom anglais et un
nom français pour
l'affichage dans le
backend

De plus, nous ajoutons un autre champ à la table dans lequel le statut du compteur sera enregistré.



Figure 7.4:
Ajout d'un champ
dans la table

Nous devons encore spécifier quelques options, comme le montre l'illustration suivante, avant que la table ne soit terminée.

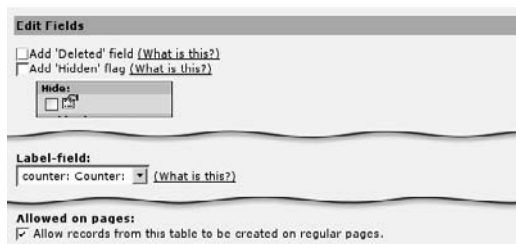
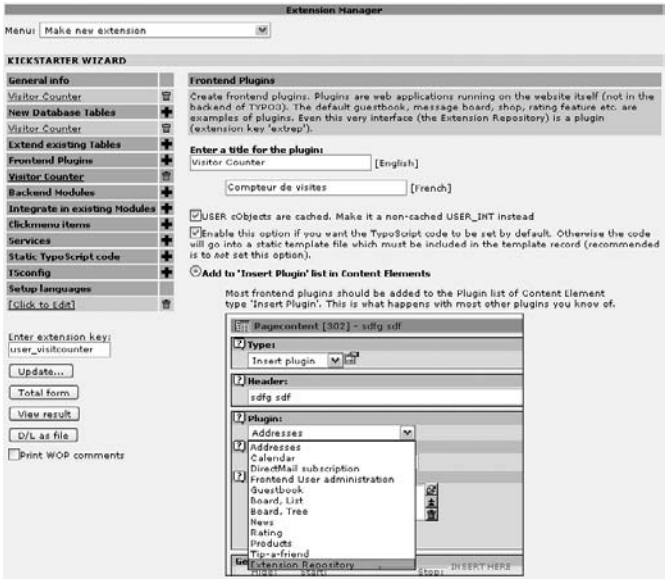


Figure 7.5:
Options de la table
autres que des
valeurs par défaut

Nous insérons maintenant un plugin dans l'extension, nous spécifions les titres en anglais et en français, et nous ajustons quelques options supplémentaires.

Figure 7.6:
Ajout du plugin et
d'autres composants
dans l'extension



Pour finir, nous ajoutons du code TypeScript statique et nous générons un aperçu du résultat avec **View Result**. L'extension peut alors être créée dans le répertoire `typo3conf/ext/user_visitcounter/` en cliquant sur **WRITE**.

Figure 7.7:
L'extension est
terminée et peut être
générée

Filename:	Size:	
ext_icon.gif	124	
ext_localconf.php	392	View
ext_tables.php	1.0 K	View
ext_tables.sql	514	View
ext_typescript_editorcfg.txt	1.0 K	View
icon_user_visitcounter.gif	135	
locallang_db.php	529	View
tca.php	1.0 K	View
doc/wizard_form.dat	2.1 K	
doc/wizard_form.html	57 K	
pi1/class.user_visitcounter_pi1.php	2.6 K	View
pi1/locallang.php	1.2 K	View
Author name:		
Author email:		
Write to location:		
Local: typo3conf/ext/user_visitcounter/ (empty)		WRITE

Le Kickstarter crée entièrement le code pour un plugin et la table de base de données. Nous pouvons directement installer cette extension avec le gestionnaire d'extensions, éditer une nouvelle table dans le backend et sélectionner le plugin frontend (qui toutefois n'affichera que des valeurs exemples).

C'est tout, du moins pour le début. En moins de quatre minutes, la base du compteur de visiteurs est fonctionnelle. Mais le plugin ne compte pas encore les visiteurs, bien sûr. 03 :39

Nous passons maintenant à un environnement de développement PHP et retirons le code et les fichiers qui ne sont pas nécessaires à ce plugin.

Nous devons effacer les données :

`pi1/locallang.php` et `ext_typoscript_editorcfg.txt`

Dans le fichier `ext_localconf.php`, les lignes de code intégrant le fichier `ext_typoscript_editorcfg.txt` sont effacées. Nous créons un fichier nommé `ext_typoscript_setup.txt` qui contiendra plus tard le code TypoScript standard pour le plugin.

Le fichier `tca.php` contient des définitions nécessaires pour pouvoir éditer la table dans le backend. Dans le Kickstarter, nous sélectionnons le type `Integer`, `10=1000` pour le champ `counter`, ce qui ne convient pas exactement à nos besoins. Nous modifierons donc la définition du champ comme suit :

```
'counter' => Array (
...
    'config' => Array (
        'type' => 'input',
        'size' => '5',
        'max' => '10',
        'eval' => 'int',
        'default' => 0
    )
),
```

Nous éditons ensuite le fichier `pi1/class.user_visitcounter_pi1.php`, qui contient le code PHP pour le plugin, et retirons le code exemple du Kickstarter. Nous pouvons maintenant commencer à configurer le compteur. 5 :20

```
require_once(PATH_tslib.'class.tslib_pibase.php');

class user_visitcounter_pi1 extends tslib_pibase {
    var $prefixId = 'user_visitcounter_pi1';    // Same as class name
    var $extKey = 'user_visitcounter';        // The extension key.

    /**
     * Main plugin function
     */
    function main($content,$conf) {
        // We do this, because it's a USER_INT object!
        $this->pi_USER_INT_obj=1;
    }
}
```

La fonction `main` sera appelée par le système frontend afin de créer le contenu du plugin.

Tout d'abord, nous devons initialiser la variable `$table` ; elle est utilisée assez fréquemment, et rend le code réutilisable.

```
$table = 'user_visitcounter';
```

Le compteur doit créer un enregistrement pour la page considérée ; nous demandons à TypoScript de trouver l'ID de la page via le paramètre `pid`. Si ce paramètre n'est pas défini, `$pid` est ajusté à l'ID de la page sélectionnée, qui est aussi disponible via `$GLOBALS['TSFE']->id`.

```

    // any page id configured?
    $pid = intval($this->cObj->stdWrap($conf['pid'], $conf['pid.']));
    // if not use current page
    $pid = $pid ? $pid : $GLOBALS['TSFE']->id;

```

Nous connaissons maintenant l'ID de la page et recherchons un enregistrement compteur. S'il n'y en a aucun, nous créons un nouveau tableau et nous initialisons le compteur à zéro.

```

    // search for counter record
    $res = $GLOBALS['TYPO3_DB']->exec_SELECTquery('', $table,
                                                'pid='.$pid, '', '', 1);
    if ($res) {
        $row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res);
    } else {
        // initialize a new counter
        $row = array();
        $row['counter'] = 0;
    }

```

Nous regardons à présent si l'utilisateur est déjà connu, et s'il ne l'est pas, le compteur passe à la valeur 1. Une vérification est effectuée pour savoir si la clé d'extension existe dans la session utilisateur. Si ce n'est pas le cas, elle est créée, et les données de la session sont réécrites, ce qui veut dire que le compteur compte les visiteurs par session.

```

    // check if this user should be counted
    $knownUser = $GLOBALS['TSFE']->fe_user->getKey('ses',
                                                $this->extKey);
    if (!$knownUser) {
        $GLOBALS['TSFE']->fe_user->setKey('ses', $this->extKey, 1);
        $GLOBALS['TSFE']->fe_user->storeSessionData();

        $row['counter'] += 1;
    }

```

L'enregistrement du compteur doit être mis à jour, ou créé s'il n'existe pas encore.

```

    // update the counter record
    if ($row['uid']) {
        $this->cObj->DBgetUpdate($table, $row['uid'], $row,
                                'counter', true);
    } else {
        $this->cObj->DBgetInsert($table, $pid, $row,
                                'counter', true);
    }

```

Pour afficher le compteur, nous créons une instance de `tslib_cObj`¹, qui est la classe principale de restitution de contenu dans le frontend et sera expliquée en détail plus loin. La méthode `cObjGetSingle()` est utilisée pour afficher le contenu en reprenant les paramètres du setup TypoScript `renderObj`. Le contenu créé est ensuite retourné.

¹La section 7.5.5 explique pourquoi une instance est créée au lieu d'utiliser `$this->cObj`.

```

        // render the counter with the TypeScript renderObj
        $lCObj = tslib_div::makeInstance('tslib_cObj');
        $lCObj->setParent($this->cObj->data,
                        $this->cObj->currentRecord);
        $lCObj->start($row, $table);
        $content = $lCObj->cObjGetSingle($conf['renderObj'],
                                       $conf['renderObj.']);

        return $this->pi_wrapInBaseClass($content);
    }
}

```

Le setup TypoScript suivant est ensuite inséré dans le fichier `ext_typoscript_setup.txt`, qui identifie l'ID de la page et définit la mise en forme du compteur avec `renderObj` : 17:04

```

plugin.user_visitcounter_pi1 {
    pid.data = page:uid

    renderObj = COA
    renderObj {
        10 = TEXT
        10.field = counter
        10.noTrimWrap = || |
        20 = TEXT
        20.value = visitors on this page
    }
}

```

Bien sûr, nous avons emprunté la voie la plus courte, et avons omis le temps de débogage. Mais cet exemple illustre tout de même comment, avec le Kickstarter et un peu d'expérience, vous pouvez obtenir rapidement des résultats.

19:46 – Terminé !

Ce compteur a la particularité de pouvoir être configuré par TypoScript. Il s'agit d'un concept général dans TYPO3, qui assure un degré de flexibilité maximum. TypoScript permet à la fois de configurer le plugin et de définir sa restitution. On peut modifier ces deux aspects, sans avoir à effectuer de changements dans le code PHP.

Le plugin est configuré en définissant l'ID de la page sur laquelle l'enregistrement du compteur doit être situé à l'aide de `pid`. Vous pourriez simplement entrer l'ID de la page en question dans le code PHP :

```
$pid = $GLOBALS['TSFE']->id;
```

Mais nous essayons plutôt d'obtenir un ID via TypoScript :

```
$pid = intval($this->cObj->stdWrap($conf['pid'], $conf['pid.']));
```

Le setup TS du `pid` est passé à la méthode `stdWrap` de l'objet `cObj`. C'est exactement la méthode que le `stdWrap` TypoScript fournit. Si vous regardez la référence TypoScript, vous verrez à quel point la gamme d'options de cet objet est étendue. Nous en ferons bon usage.

Les deux lignes de code suivantes donnent le même résultat :


```
// PHP:
$pid = $GLOBALS['TSFE']->id;

// TypoScript:
pid.data = page:uid
```

En utilisant TypoScript pour la configuration, le plugin peut être utilisé de différentes façons. Vous pouvez par exemple écrire :

```
pid = 12
```

Le plugin ici ne comptera pas le nombre de visiteurs par page, mais le nombre de visiteurs pour tout le site, en n'enregistrant le statut du compteur que sur la page ayant l'ID=12. Vous devriez alors modifier aussi le texte affiché par le plugin. Les setups TS nécessaires à cet ajustement seraient alors les suivants :

```
plugin.user_visitcounter_pil {
    pid = 12
    renderObj.20.value = visitors on this website
}
```

Il est donc possible, avec l'aide de TypoScript, de modifier la fonctionnalité des plugins dans une certaine mesure (pour autant que vous l'utilisiez dans le code PHP).

Vous pouvez encore faire d'autres modifications, par exemple traduire le titre du compteur en français. Pour ce faire, redéfinissez `renderObj.20.value`, ou ajoutez le texte en français comme une option.

```
plugin.user_visitcounter_pil {
    pid = 12
    renderObj.20.value = visitors on this website
    renderObj.20.lang.fr = Visiteurs de ce site
}
```

Si le site Web est correctement configuré pour la langue française, le texte français est alors affiché automatiquement.

```
config.language = fr
```

Bien sûr, il n'est pas très pratique de devoir activer le compteur sur chaque page avec l'élément de contenu **Insérer un plugin**, même s'il compte les visiteurs pour l'entièreté du site. Ceci peut être évité en insérant le plugin en un point approprié du gabarit du site.

```
page.80 =< plugin.user_visitcounter_pil
```

Le plugin est maintenant activé sur chaque page.

Puisque le plugin n'effectue pas lui-même la restitution du compteur, mais ne fait que passer les réglages TypoScript de `renderObj` à `cObjGetSingle()`, vous pouvez utiliser toutes les options offertes par TypoScript en ce qui concerne l'affichage. Le compteur de visiteurs peut donc bénéficier des fonctions graphiques.

Pour afficher le compteur comme à la figure suivante, vous ne devez effectuer aucune modification dans le code PHP, seulement quelques adaptations TypoScript.



Figure 7.8:
Exemple de compteur

```
plugin.user_visitcounter_pi1.renderObj >

plugin.user_visitcounter_pi1.renderObj = IMAGE
plugin.user_visitcounter_pi1.renderObj {
    alttext.field = counter
    file = GIFBUILDER
    file {
        XY = [10.w]+20,27
        backColor = #000000
        10 = TEXT
        10 {
            text.field = counter
            fontSize = 21
            fontFile = fileadmin/Facelift.ttf
            fontColor = #00EE00
            offset = 6,22
        }
    }

    20 = BOX
    20.dimensions = 0,0,200,1
    20.color = #00EE00
    21 < .20
    21.align = , b
    22 < .20
    22.dimensions = 0,0,1,40
    23 < .22
    23.align = r
}
}
```

Notez que le compteur de visiteurs n'est utilisé ici qu'à titre d'exemple. Il existe certainement de meilleurs compteurs que celui-ci. Créer le compteur avec un habillage n'a servi ici qu'à illustrer le lien entre les langages PHP et TypoScript. Si le compteur avait réellement été utilisé de cette façon, le serveur aurait généré un total de 135 487 fichiers image, qui devraient un jour être supprimés.

7.2 Assistant d'extensions : le Kickstarter

Comme l'exemple du compteur de visiteurs l'a illustré, le Kickstarter crée un framework complet dans lequel il reste à ajouter les fonctionnalités. Toutes les extensions créées par le Kickstarter peuvent être installées immédiatement, et affichent déjà un « Hello world ». Une extension enregistrée peut être rechargée et éditée par le Kickstarter, mais le code déjà entré manuellement sera perdu lors de l'enregistrement, car le Kickstarter n'est pas un éditeur pour

les extensions existantes ; il sert seulement à en démarrer une nouvelle. Avant de commencer à créer une extension, vous devez choisir une clé d'extension.

7.2.1 Définition d'une clé d'extension

Chaque extension a son propre espace de nommage, défini par la clé d'extension. Cette clé représente une courte chaîne de caractères pouvant comporter les caractères de a à z, de 0 à 9, ainsi que « _ ». Il existe deux types d'extensions, qui diffèrent par leur clé : d'un côté les extensions normales, pouvant aussi être publiées dans le répertoire d'extensions TYPO3 (TER), et de l'autre les extensions spécifiques à un projet, qui ne peuvent pas être envoyés vers le répertoire d'extensions. On les reconnaît grâce au préfixe `user_` dans la clé d'extension. Les extensions normales peuvent avoir n'importe quelle clé ne commençant pas par `tx` ou `u`.

Référence 104717

Alors que les clés spécifiques à un projet utilisant le préfixe `user_` peuvent porter n'importe quel nom, ce n'est pas le cas pour les extensions normales. Celles-ci doivent être enregistrées afin d'être uniques (cf. référence ci-contre). Après cet enregistrement, vous disposez de dix jours pour changer l'extension dans le répertoire, sinon, la clé sera effacée. Si vous n'avez pas complété votre extension dans ce délai, vous pouvez toujours la transférer de nouveau vers le répertoire, pour éviter que la clé ne soit perdue. Après enregistrement, l'extension est d'abord marquée comme **Members only**, ce qui signifie qu'après l'avoir téléchargée, l'extension n'est pas visible pour les autres utilisateurs, s'ils ne sont pas identifiés en tant que membres.

Il est recommandé de choisir un nom explicite pour l'extension, aussi court que possible : explicite, pour que vous (et les autres) puissiez reconnaître de quel type d'extension il s'agit à partir du nom du répertoire de l'extension, et court, parce que le code du programme utilise cette clé comme espace de nommage. Une clé ayant un long nom impliquerait de longs noms de variables dans les codes PHP, HTML et CSS. Vous pouvez utiliser les traits de soulignement « _ », mais le Kickstarter les enlève pour certains noms, ce qui peut prêter à confusion. La solution la plus simple est donc d'éviter les traits de soulignement dans le nom des clés. Une fois que vous avez défini une clé d'extension, saisissez-la dans le champ situé dans le coin inférieur droit du Kickstarter.

7.2.2 Composants de Kickstarter

La figure 7.9 illustre une extension comportant tous les composants pouvant être générés par le Kickstarter. Vous pouvez aussi y voir quels fichiers ont été créés lors du processus.

Les composants peuvent être ajoutés individuellement avec les icônes +, et retirés avec l'icône « Corbeille ». Si une extension contient plusieurs composants du même type, les répertoires sont numérotés en conséquence : `mod1/`, `mod2/`.

Les différents composants sont les suivants :

General info

Toutes les extensions doivent normalement contenir le composant **General info**. Il définit de l'information générale concernant l'extension, qui sera affichée dans le TER (répertoire d'extensions TYPO3), et dans le gestionnaire d'extensions (EM).

New Database Tables

Sert à créer de nouvelles tables de base de données. À l'aide d'un assistant, vous pou-

vez définir des champs et des types de champs (chaîne de caractères/champ de saisie, nombre entier/case à cocher, etc.).

Extend existing Tables

Ce composant permet d'étendre des tables de base de données déjà existantes. Le même assistant est disponible pour les définitions de champs.



Figure 7.9:
Exemple d'extension
utilisant tous les
composants pouvant
être créés dans le
Kickstarter

Frontend Plugins

Ajoute un plugin frontend à l'extension. Les plugins peuvent étendre le frontend de plusieurs façons. On peut par exemple créer de nouveaux éléments de contenu, des menus, ou des applications frontend. Le Kickstarter propose un choix parmi les plugins les plus courants.

Backend Modules

Cette option ajoute un module backend à l'extension. On peut l'insérer dans différents modules principaux tels que **Web**, **Fichier** ou **Outils**.

Integrate in existing Modules

Certains modules peuvent être étendus par des fonctions de sous-module. Le Kickstarter les supporte dans une série de modules **Web** ainsi que pour le module **Utilisateur** → **Centre de tâches**.

Clickmenu items

Permet de générer des entrées dans le menu contextuel.

Services

Crée un composant service.

Static TypeScript code

Ce composant permet d'ajouter du TypeScript à l'extension. Après avoir installé l'extension, le code TS est actif dans tout le système.

TSConfig

Permet d'ajouter une configuration TS utilisateur et page.

Setup languages

Les plugins et les modules fournissent tous deux du support pour plusieurs langues. Dans le Kickstarter, les textes (identifiants de champs, titres, etc.) peuvent être définis dans différentes langues, si elles sont ajoutées avec ce composant.

7.2.3 Structure d'une extension

Référence 099886

Comme décrit précédemment, les extensions consistent en une série de fichiers et, éventuellement, de répertoires. Ces fichiers remplissent des fonctions spécifiques déterminées suivant leur nom, ou bien sont inclus par l'API de l'extension.

Une extension étant normalement créée à l'aide du Kickstarter, elle contient donc déjà tous les fichiers et les répertoires dans la forme appropriée. Nous ne donnerons donc ici qu'un aperçu de leur contenu et de leurs fonctions. Les détails sont expliqués dans les exemples lorsque c'est nécessaire. De plus, nous vous signalons la référence ci-contre, qui contient la documentation appropriée.

ext_emconf.php

Contient toutes les informations générales et les données méta concernant l'extension. Il s'agit généralement de données provenant du composant **General info** du Kickstarter.

ext_localconf.php

Pour chaque requête, ce fichier est inséré à la fois dans le backend et le frontend, et contient toutes les configurations de type (\$TYPO3_CONF_VARS) et de type `typo3conf/localconf.php`. De plus, du code TypeScript peut aussi être inclus via l'API de l'extension.

ext_tables.php

Contient la configuration des tables de base de données destinées à l'usage du backend. Les plugins et les modules sont inclus via l'API de l'extension. Le code correspondant est généré par le Kickstarter et doit rarement être modifié.

ext_tables.sql

Ce fichier contient les données SQL pour la définition des tables. Cette définition est

reconnue et analysée par l'EM et l'outil d'installation. De cette façon, l'EM (gestionnaire d'extensions) et l'outil d'installation peuvent détecter si une table doit être créée ou mise à jour.

ext_tables_static+adt.sql

Contient des définitions de tables SQL, y compris les données ; il est prévu pour les données statiques, comme on les retrouve par exemple dans l'extension **Static Info Tables**. Cette extension contient des données à propos des pays, des langues et des devises. L'abréviation « adt » signifie « add drop table » et se réfère à l'option du même nom, utilisée lorsque la table est créée avec l'outil **mysqldump**, de façon à ce que les instructions SQL correspondantes puissent être placées au début du code SQL.

```
DROP TABLE IF EXISTS static_countries;
```

Pour être reconnue correctement, la définition de la table doit aussi exister dans **ext_tables.sql**.

ext_typoscript*.txt

Contient du code TypoScript intégré globalement, pour qu'il ne soit pas seulement disponible via les enregistrements de gabarits (cf. **static/**).

ext_conf_template.txt

Ce fichier maintient si nécessaire les définitions des options de configuration des extensions. En fonction de ces définitions, ces options sont visibles pour l'utilisateur dans l'EM.

***icon*.gif**

Contient les fichiers icônes pour l'extension, les tables de base de données, les plugins, les modules, etc.

locallang*.php

Ces fichiers contiennent des textes écrits dans les différentes langues utilisées par les plugins ou par les modules, ainsi que les définitions TCA.

class*.php

Contient les classes PHP pour les plugins et les fonctions de sous-modules.

class.ext_update.php

Autorise les fonctions permettant de mettre à jour l'extension dans l'EM ; l'extension **newloginbox** en fournit un exemple.

conf.php

Ce fichier configure un module et l'insère dans le backend.

index.php

Le fichier **index.php** constitue généralement le script principal d'un module.

pi1/

Ce répertoire contient les scripts et les données d'un plugin.

`cm1/`

Ce répertoire contient les scripts et les données des menus contextuels.

`mod1/`

C'est le répertoire d'un module. Il contient les fichiers `conf.php` et `index.php`.

`modfunc1/`

Contient les scripts et les données des fonctions de sous-modules.

`static/`

Ce répertoire contient des fichiers gabarits TypoScript insérés via l'API de l'extension, et qui sont ensuite disponibles dans les enregistrements de gabarits, de la même façon que les gabarits statiques.

`sv1/`

Répertoire contenant des services.

`res/`

Répertoire pour toutes sortes de données (« ressources »).

`doc/`

Répertoire pour la documentation ; contient aussi le fichier `wizard_form.dat`, dans lequel se trouve l'ensemble de la définition générée par le Kickstarter. Il est possible de créer une nouvelle extension dans le Kickstarter à partir de cette information, sur base d'une extension existante.

7.2.4 Règles de base des extensions

Composants

Comme on l'a déjà mentionné, une extension peut avoir plusieurs composants. Toutefois, toutes les combinaisons n'ont pas de sens : par exemple, une extension ne devrait contenir que les composants d'une seule application. Une extension pour effectuer une réservation est un bon exemple d'extension comportant plusieurs composants : des tables de base de données, un plugin pour afficher les données dans le frontend et permettre de faire des réservations, ainsi qu'un module backend pour l'affichage et l'administration des réservations. Il ne serait pas logique de lui ajouter un plugin affichant les actualités de l'office du tourisme de votre localité. Il s'agit d'une fonctionnalité distincte, qui devrait constituer une extension séparée. On applique la règle suivante : à moins qu'il ne soit réellement utile d'installer tous les composants dans une même extension, ils devraient avoir chacun leur propre extension.

Documentation

Même s'il n'est pas toujours possible de fournir une documentation complète avec une extension, elle devrait au moins contenir un manuel (`doc/manual.sxw`) comportant la section « Introduction ». Cette section doit contenir une description, et si possible une capture d'écran. Pour un plugin, il est important de préciser s'il s'agit d'un élément de contenu, par exemple, ou s'il doit être intégré dans une page à l'aide d'**Insérer un plugin**.

En général, il n'est pas nécessaire que cette description soit particulièrement longue : elle peut ne fournir qu'un aperçu. La section « Introduction » étant affichée dans la vue détaillée de l'extension dans le TER de TYPO3, vous pouvez alors déjà avoir une bonne idée de ce qu'est l'extension. Cette information vous permet de juger si l'extension convient à l'usage que vous souhaitez en faire. Pour quelqu'un qui recherche une extension en particulier, cela lui évite de perdre du temps à essayer différentes extensions.

Notez qu'un manuel transféré vers le répertoire d'extensions pour la première fois, comme décrit à la section 7.3.5, doit d'abord être initialisé. Sinon, la section « Introduction » ne sera pas reconnue.

Catégories

Il est important que les extensions soient classées dans les bonnes catégories. Elles doivent être classées dans les catégories de base (backend, modules backend, frontend, etc.) utilisées pour le classement dans l'EM, mais aussi dans les catégories du TER liées à l'application, que vous pouvez assigner à l'extension sur TYPO3.org (cf. section 7.3.4).

Alors que les catégories de base permettent de garder un affichage clair dans l'EM, les catégories du TER permettent de trouver l'extension appropriée en fonction d'un usage particulier.

Publication

Il est important de mentionner encore une fois que toutes les lignes de code utilisées dans le cadre de TYPO3 (qui correspondent à l'extension que vous avez vous-même écrite) sont automatiquement soumises à la licence GPL. Cela ne signifie pas que vous devez publier votre extension, mais gardez à l'esprit que plusieurs années de travail ont été consacrées à ce projet, et que toute contribution permettant à TYPO3 de progresser vous est aussi profitable.

7.3 Gestion d'extensions pour les programmeurs

7.3.1 Fonctions du gestionnaire d'extensions

Le gestionnaire d'extensions propose plusieurs fonctions intéressantes qui sont très utiles à l'administration de vos propres extensions. On retrouve ces fonctions dans la vue détaillée d'une extension, obtenue en cliquant sur le nom d'une extension listée dans l'EM. Un menu s'affiche alors, contenant différentes fonctions.



Figure 7.10:
Fonctions dans la vue
détaillée de l'EM

Information

De l'information détaillée à propos de l'extension est reprise ici. Hormis les détails généraux tels que le nom, la description et le numéro de version de l'extension, on y retrouve de l'information sur les fichiers et les tables de base de données. On y voit aussi éventuellement les erreurs relatives à l'extension considérée ; comme une violation de l'espace de nommage de l'extension, ou des définitions XCLASS manquantes (cf. section 7.10). Les messages d'erreur sont marqués en rouge.

Edit files

Vous permet d'éditer en ligne les fichiers de l'extension ; ce qui peut parfois être utile pour corriger de petites erreurs sur le serveur Web.

Backup/Delete

Un certain nombre de fonctions sont regroupées dans cet élément, comme vous pouvez le constater à la figure suivante. Celles-ci sont décrites dans les paragraphes suivants.

Figure 7.11:
Fonctions du menu
Backup/Delete dans
l'EM



Backup → Extension files

Il est certainement très utile de pouvoir télécharger l'extension en tant que fichier T3X. Ce paquetage contient toute l'extension, et peut être réinstallé grâce à l'EM. De cette façon, il est possible d'envoyer les extensions simplement par email, sans avoir à compresser manuellement tous les fichiers nécessaires dans une archive zip. De plus, il arrive fréquemment que l'extension PHP pour la décompression de fichiers ne soit pas disponible, en particulier sur les serveurs Windows. Ainsi, certains paquetages risquent de demeurer compressés. On reconnaît les paquetages au `-z` apparaissant à la fin du nom du fichier `*-z.t3x`.

Backup → Data tables

Cette fonction permet de télécharger la table de base de données de l'extension, si elle existe.

Delete

Cette fonction efface complètement l'extension du serveur, à condition qu'elle ne soit pas active.

Update EM_CONF

Vous permet de mettre à jour le fichier `ext_emconf.php` ou une extension. Ce fichier fait le compte, en quelque sorte, des versions des fichiers de l'extension, ce qui aide à vérifier si les fichiers ont été modifiés, rendant nécessaire une mise à jour de la version dans le TER. Cette information est affichée en conséquence dans l'aperçu disponible sous **Information**. Il est recommandé d'utiliser cette fonction avant de transférer une extension vers le TER.

Make new extension

Cette fonction vous permet d'utiliser l'extension comme gabarit, ou de créer une nouvelle extension dans le Kickstarter. On peut y spécifier une nouvelle clé d'extension, puis développer la nouvelle extension. Mais cela ne signifie pas que la fonctionnalité de cette extension est reprise dans la nouvelle extension, car le Kickstarter crée un nouveau framework qui ne reprend pas le code spécifiquement écrit pour l'extension sélectionnée. Cette fonction est utile, par exemple, si vous souhaitez créer une extension ayant une table de base de données identique ou similaire à celle d'une extension préexistante.

Dump DB

Vous pouvez l'utiliser pour mettre à jour la définition de la base de données dans `ext_tables.sql`. Cette fonction est utile si, par exemple, vous avez effectué des modifications dans la définition de tables à l'aide de `phpMyAdmin`, et que vous désirez les insérer dans l'extension. N'oubliez pas que les définitions de la table du TCA ont peut-être besoin d'être adaptées.

Upload

Cette fonction permet de transférer l'extension vers le TER. Elle est décrite plus en détail dans les sections qui suivent.

7.3.2 Compte utilisateur TER

Avant de pouvoir télécharger des extensions vers le TER (sur TYPO3.org), vous devez d'abord créer un compte utilisateur sur TYPO3.org (cf. référence ci-contre). Ensuite, saisissez votre nom d'utilisateur et votre mot de passe dans l'EM sous **Settings**. Nous vous signalons que l'EM, seulement avec ces données d'accès, peut afficher les extensions pour lesquelles vous êtes considéré comme membre (**Members only**).

Référence 900162

Figure 7.12:
Saisie des données
d'accès au répertoire
TER dans l'EM

Un mot de passe est nécessaire au téléchargement d'une extension. Il peut varier d'une extension à l'autre. Si vous utilisez souvent le même mot de passe, vous pouvez entrer un mot de

pas standard dans le champ **default upload password**. Le mot de passe est défini lors de l'enregistrement de la clé d'extension, mais peut être modifié par la suite.

7.3.3 Transfert d'une extension vers le TER

Pour transférer une extension vers le TER, vous devez accéder à la vue détaillée en cliquant sur le nom de l'extension en question (dans la liste fournie par le gestionnaire d'extensions). Sélectionnez ensuite le menu **Upload** afin d'afficher le formulaire illustré à la figure suivante.

Figure 7.13:
Préparation d'une
extension afin qu'elle
soit transférée dans
le répertoire
d'extensions en ligne

Extension: FE Debug/Info output (cc, feinfo) [Upload] [Go back]

UPLOAD EXTENSION TO REPOSITORY

Repository:

Username:

Repository Password:

Upload password for this extension:

Comment to the upload:

Upload command:

- ☐ New development version (latest x.x.x = 1)
- ☐ Override this development version (0.1.0)
- ☐ New sub version (latest x.x = 1.0)
- ☐ New main version (latest x = 1.0.0)

☐ Yes, don't show this upload in the public list.
(“Private” uploads requires you to manually enter a special key (which will be shown to you after the upload has been completed) to be able to import and view details for the upload. This is nice when you are working on something internally which you do not want others to look at.)

Private? ☐

Additional import password:

(Anybody who knows the “special key” assigned to the private upload will be able to import it. Specifying an import password allows you to give away the download key for private uploads and also require a password given in addition. The password can be changed later on.)

9.7 K, compressed, base64

Clicking “save as file” will allow you to save the extension as a file. This provides you with a backup copy of your extension which can be imported later if needed. “Save as file” ignores the information entered in this form!

Normalement, il vous suffit de sélectionner la procédure à utiliser avec le numéro de version de l'extension. Elle peut être directement incrémentée par le transfert. Cependant, l'incrémentation du numéro de version ne signifie pas qu'elle sera affichée dans l'EM ou le TER en tant que nouvelle version. Cette action peut être utile si, par exemple, vous devez apporter de légères modifications à la documentation, et qu'il ne vaudrait pas la peine que les utilisateurs la téléchargent de nouveau. Si toutefois une erreur a été rectifiée, vous devrez sélectionner le chiffre au centre, afin de mettre à jour le numéro de version.

Si vous avez choisi un mot de passe de transfert spécial pour l'extension sélectionnée, vous devez le saisir ici. Sinon, la valeur **default upload password** définie dans **Settings** sera automatiquement utilisée.

Si une extension est transférée pour la première fois vers le TER, elle est marquée en tant que **Members only**, et n'est donc accessible que pour les membres identifiés. L'option **Private** propose une façon de fournir la version courante de votre extension avec un mot de passe. Cette option est très utile si vous voulez faire tester une nouvelle version d'une extension déjà disponible par certaines personnes, sans la rendre publique.

En cliquant sur **Upload extension**, vous transférez l'extension. Quand la procédure est terminée, un message apparaît. Le numéro de version ayant été incrémenté dans le répertoire d'extension, il doit bien sûr l'être aussi dans la version locale. La mise à jour doit encore être confirmée après que le numéro de version a été modifié. Le transfert vers le TER est alors terminé.

7.3.4 Gestion d'extensions TER

Vous avez la possibilité de gérer vos extensions dans le TER sur TYPO3.org. Pour ce, vous devez d'abord vous identifier. Si vous sélectionnez ensuite un élément du menu principal, vous pouvez accéder à vos propres extensions via l'option additionnelle **My extension keys**.

Référence 156502

Une liste s'affiche, comme illustré à la figure suivante. Vous pouvez classer vos extensions selon divers critères, et afficher quelques données statistiques. Si vous cliquez sur une des entrées de la liste, un formulaire apparaît, donnant quelques détails à propos de l'extension, et proposant quelques options de configuration.



Figure 7.14:
Liste de vos propres extensions sur TYPO3.org

Vous pouvez, à cet endroit, modifier le nom et la description de l'extension. L'extension peut aussi être classée dans de nouvelles catégories, que vous pouvez visualiser dans la vue **Catégories** du TER. Les versions transférées jusqu'à présent y sont listées. Vous pouvez effacer les anciennes versions ; ce qui devrait être fait régulièrement, afin de libérer de l'espace de mémoire dans le répertoire.

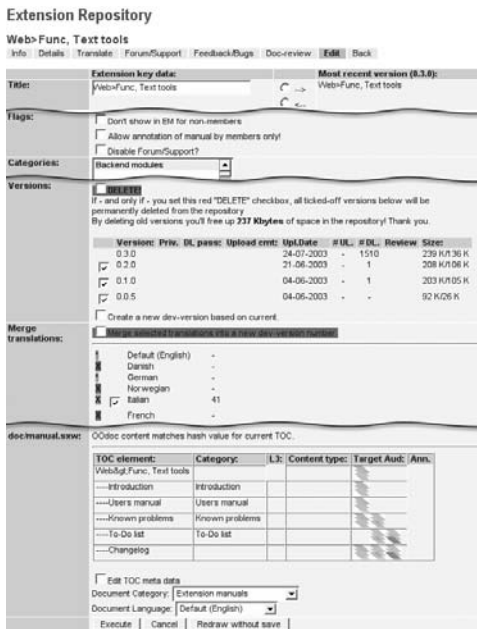


Figure 7.15:
Information détaillée et configuration pour une extension

Si l'extension contient des fichiers de langues `locallang`, vous aurez aussi un aperçu des traductions vers d'autres langues. Cette étape achevée, vous pouvez ensuite configurer le répertoire contenant la documentation. Vous trouverez de plus amples explications à la section suivante.

Members only

La restriction de l'accès aux membres constitue une fonction importante : elle est très utile pendant la phase de développement, pour permettre seulement à certains testeurs d'avoir accès aux versions beta des extensions. Comme mentionné plus haut, toute extension est d'abord marquée comme **Members only**. Pour ajouter des membres à une extension, saisissez leurs noms, ou leurs noms d'utilisateur dans TYPO3.org, dans le champ prévu à cet effet dans **Lookup users**. Si vous soumettez le formulaire, tous les utilisateurs répondant à cette description seront listés ; vous n'avez alors plus qu'à choisir.

Afin de rendre une extension disponible pour le grand public, c'est-à-dire de la publier dans le TER, vous devez désactiver l'option **Members only**. Vous ne pouvez toutefois pas la réactiver : une fois qu'une extension est publiée, elle le demeure. Il est possible de garder cachées les versions individuelles d'une extension déjà publiée, si celles-ci sont transférées avec l'option **Private**, ce qui ne les rend accessibles qu'aux utilisateurs munis d'un mot de passe approprié.

Figure 7.16:
Saisie des membres
d'une extension

The screenshot shows a web form for managing extension members. On the left, there's a grey sidebar with the text 'Members only:' and 'Members:'. The main area on the right contains a list of members, each with a checkbox and their details. The members listed are:

- ☒ daniel (Daniel Hinderink, daniel@typo3.com)
- ☒ werner (Werner Altmann, wa@artplan21.de)
- ☐ kasper (Kasper Skårhøj, kasper@typo3.com)

 Below the list is a section labeled 'Lookup users:' with a text input field containing the name 'kasper'.

7.3.5 Publication de documentation

Chaque extension devrait avoir sa propre documentation dans le format spécifié. Cela ne signifie toutefois pas qu'une extension ne peut pas être publiée en l'absence de documentation. En cas de doute, l'esprit du projet consiste à publier un travail non documenté mais concret, plutôt que de perdre trop de temps à essayer d'atteindre un très haut niveau de qualité.

La documentation d'une extension se trouve dans le fichier `manual.sxw` du sous-répertoire `doc/`. Ce fichier contient le manuel en format OpenOffice, et est transféré avec l'extension vers le TER. Ce système de documentation n'est pas seulement utilisé pour les extensions comportant des modules et des plugins, mais aussi pour la documentation générale. Les extensions de type documentation contiennent seulement les fichiers `ext_emconf.php` et `ext_icon.gif` en plus du fichier `doc/manual.sxw`.

Si vous n'avez pas encore OpenOffice, vous devriez l'installer maintenant (téléchargez-le à partir de <http://www.openoffice.org>). Il s'agit d'un paquetage bureautique très puissant, disponible gratuitement pour les plate-formes Unix/Linux, OS X et Windows. L'avantage du format OpenOffice est qu'il est ouvert. Le format est documenté et disponible en tant que XML, ce qui permet à d'autres logiciels de le lire et de le convertir facilement. C'est aussi la raison pour laquelle la documentation s'affiche automatiquement sur le site de TYPO3.org après que vous avez téléchargé votre extension.

Pour documenter une extension, certaines conditions doivent être remplies. Vous devez d'abord télécharger le gabarit du manuel à partir de TYPO3.org (cf. référence ci-contre) et l'enregistrer dans le répertoire `doc/` sous le nom `manual.sxw`. Le gabarit contient déjà des sections explicatives.

Référence 167050

Si nécessaire, vous pouvez ajouter de nouvelles sections à la documentation. Compte tenu du fait que chaque section n'est pas indispensable à chaque extension, votre manuel ne doit pas obligatoirement reprendre toutes les sections du gabarit. À la fin du gabarit se trouve une section contenant des indications générales pour la création de documentation avec OpenOffice, qui devraient être entièrement effacées. La référence ci-contre donne une courte version du guide « Writing documentation for TYPO3 ».

Référence 600297

Lorsque vous avez terminé votre documentation, vous la rendez disponible en transférant l'extension vers le TER. Il est recommandé d'y insérer au moins la section « Introduction », qui donne une description générale de l'extension. Après le premier transfert – ou si vous lui avez ajouté de nouvelles sections – vous devez initialiser ou mettre à jour la table des matières sur TYPO3.org. Pour ce faire, affichez la vue détaillée de l'extension dans le répertoire, en sélectionnant l'option **Édit TOC meta data** dans la section `doc/manual.sxw`. Après avoir soumis le formulaire, vous pouvez adapter la table des matières. Les sections provenant du gabarit sont reconnues automatiquement. La langue dans laquelle le document a été écrit doit être assignée en conséquence. Même si la langue standard de la documentation au sein du projet TYPO3 est l'anglais, vous avez aussi la possibilité d'écrire votre documentation dans d'autres langues.

Pour chaque section, vous pouvez aussi définir le groupe cible pour lequel elle présentera un intérêt : rédacteurs, administrateurs ou développeurs. On y parvient en sélectionnant les icônes représentant différentes couches mises en évidence. Vous avez certainement remarqué qu'il s'agissait du logo TYPO3. Ces niveaux représentent les trois rôles utilisateur ; le premier niveau (le plus proche de la surface) représente le rédacteur, celui du centre, l'administrateur, et le plus bas (le plus proche du noyau), le développeur.

doc/manual.sxw: ODoc content matches hash value for current TOC.

TOC element:	Category:	L3:	Content type:	Target Aud:	Ann.
Web>Func. Text tools				☒ ☐ ☐	
----Introduction	Introduction	☐		☒ ☐ ☐	
----Users manual	Users manual	☐		☒ ☐ ☐	
----Known problems	Known problems	☐		☒ ☐ ☒	
----To-Do list	To-Do list	☐		☒ ☐ ☒	
----Changelog		☐		☒ ☐ ☒	

☒ Edit TOC meta data
Document Category:
Document Language:

Figure 7.17:
Édition de la structure
du contenu d'un
manuel d'extension

7.4 Le framework TYPO3

TYPO3, avec son architecture, représente un framework qui peut être étendu efficacement par des extensions. La modularité du framework permet de maintenir les principes de l'architecture malgré son haut degré de fonctionnalité. Cette capacité d'expansion est rapidement disponible à l'aide de l'*Extension Kickstarter*, qui a déjà convaincu de nombreux développeurs, comme l'atteste le nombre élevé d'extensions publiques déjà créées. L'effet de synergie causé par l'utilisation combinée des fonctions rendues disponibles par le cœur du système et par d'autres extensions permet d'effectuer de nouveaux développements d'un niveau significativement plus haut que ceux basés sur des solutions autonomes.

Nous allons fournir une vue d'ensemble du framework TYPO3 ci-dessous. Nous aborderons les concepts, les modèles de données, les fonctions du cœur du système ainsi que des notes générales sur la programmation. Certaines remarques, du point de vue de la programmation dans le frontend ou le backend, sont reprises dans les sections qui suivent.

Donner une vue d'ensemble ne signifie pas remplacer la documentation et les références existantes—ce qui n'aurait de toute façon aucun sens. Vous devrez toujours garder un œil sur le matériel de référence, pour examiner les détails des fonctionnalités et des options. Mais nous pensons que cette vue d'ensemble vous fournira un bon point de départ pour comprendre la logique du système.

7.4.1 Structure du framework

TYPO3 n'est pas monolithique : sa structure est modulaire. À tel point qu'à la fois le frontend et le backend peuvent être remplacés ou supprimés. Même si plus de 99 pour cent des installations TYPO3 sont utilisées comme CMS, le système est en premier lieu un système de gestion de bases de données, qui repose sur les technologies du Web.

Le cœur de TYPO3 se définit essentiellement par un modèle de base de données et de fonctions d'accès correspondantes dans le fichier `class.t3lib_tcemain.php` (TCE). Dans ce modèle, vous pouvez organiser n'importe quelle table de base de données dans un type de répertoire similaire dans sa structure à un système de fichiers. L'administration des utilisateurs et de leurs droits fait déjà partie du système de base.

Le CMS utilise ce modèle de données pour la gestion de la structure des pages avec leur contenu. Cela signifie que le frontend, ainsi que les tables de base de données associées (`tt_content`, ...) et les types de pages (**Extended, Not in the menu**, ...) représentent une application (une extension) du framework TYPO3. Il est possible par exemple de développer un logiciel intranet de comptabilité sur base du backend TYPO3, sans même installer le frontend du CMS.

Un autre scénario envisageable est d'avoir un frontend spécialisé très performant, qui n'est pas structuré selon la technique TypoScript, mais qui produit tout de même immédiatement un résultat (HTML, XML, ...).

Même le backend est remplaçable, comme nous l'avons prouvé en pratique. Si vous n'utilisez pas depuis longtemps TYPO3, vous ne connaissez probablement pas le *Classic Backend*. Ce backend était une application JavaScript assez impressionnante mais, de l'aveu général, un peu lente et difficile à maintenir.

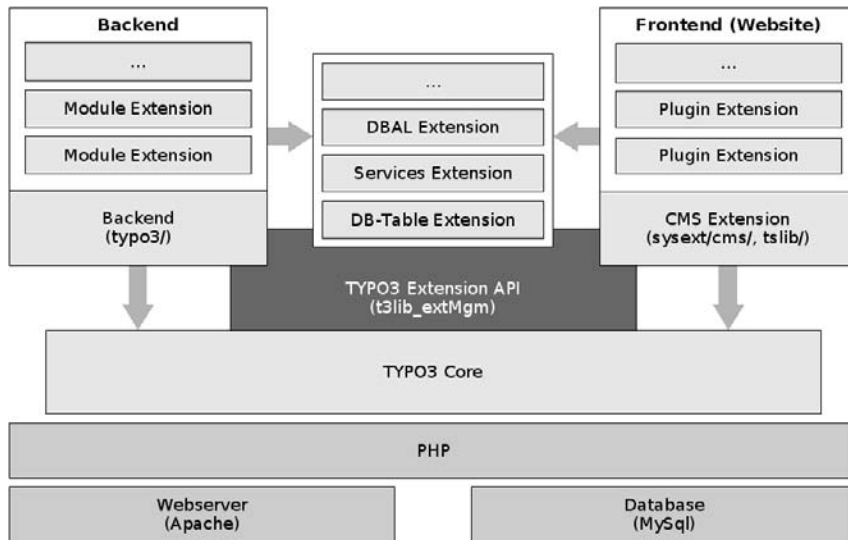


Figure 7.18:
Structure du
framework TYPO3

Comme vous le savez déjà, TYPO3 peut s'étendre simplement à l'aide des extensions. Le CMS lui-même est une extension. Du point de vue du framework, les extensions se divisent fondamentalement en cinq catégories :

- Basées uniquement sur le cœur de l'application
- Basées sur le backend standard (typo3/)
- Basées sur le frontend du CMS (sysext/cms/)
- Basées sur un autre backend
- Basées sur un autre frontend

Les trois premières catégories se mélangent fréquemment les unes aux autres : par exemple, dans une extension avec une table de base de données (le cœur) et un plugin approprié (le CMS) pour l'affichage.

Les deux dernières catégories sont bien plus rares, puisque TYPO3 est généralement utilisé comme un CMS avec un backend et un frontend standards.

7.4.2 Conventions d'écriture

Avant de considérer le framework de plus près, nous devrions introduire brièvement les *conventions d'écriture du code* adoptées par le projet TYPO3. Elles vous aideront en particulier si vous êtes habitué à intégrer du code PHP directement dans les documents HTML, car cette pratique n'est presque plus utilisée dans TYPO3.

Référence 485098

Ainsi, du code sous la forme :

```
<?php for ($a=1;$a<=$count;$a++) { ?>
    <input type="file" name="upload_<?php echo $a ?>" size="50" /><br />';
<?php } ?>
```


n'apparaît *jamaïs* dans TYPO3. Au lieu d'intégrer du code PHP dans de l'HTML, avec TYPO3, nous faisons exactement l'inverse :

```
for ($a=1;$a<=$count;$a++) {  
    $code.= ' <input type="file" name="upload_'. $a.'" size="50" /><br />';  
}  
...  
echo $code
```

Cette procédure est peut-être nouvelle pour vous : la première possibilité est encore beaucoup utilisée, surtout dans les petits projets, même si la seconde est écrite plus clairement et s'exécute en outre plus rapidement.

En général, dans le projet TYPO3, l'accent est mis sur le respect des conventions suivantes par le code PHP et HTML :

- compatibilité avec XHTML (transition) et CSS
- utilisation de simples guillemets anglais pour les chaînes de caractères
- utilisation de `htmlspecialchars()`, `$GLOBALS['TYPO3_DB']->quoteStr()` et de `intval()` pour empêcher les attaques de type *cross-site scripting*
- documentation complète des classes et des fonctions à l'aide de commentaires Javadoc/PHPDoc

Presque tout le code TYPO3 est inséré dans des classes. Seules quelques fonctions et variables globales sont présentes dans le code. Cette pratique doit aussi être respectée dans les extensions, car c'est le seul moyen de garantir la sécurité et la stabilité du système lorsque plusieurs dizaines d'extensions sont installées. Les classes sont instanciées sous la forme d'objets, à quelques exceptions près, comme par exemple pour la classe de `class.t3lib_div.php` qui sert uniquement d'espace de nommage pour les fonctions qu'elle contient.

De l'information plus détaillée sur les conventions d'écriture est disponible à la référence de cette section. Gardez à l'esprit que la mise en forme du code dans ce livre ne respecte pas strictement les conventions d'écriture, à cause de l'espace d'impression limité. Tous les exemples peuvent être téléchargés et ceux-là sont mis en forme correctement.

7.4.3 Structure des répertoires

Ci-dessous, vous trouvez une vue d'ensemble de la structure des répertoires de TYPO3, parce qu'en allant regarder le code source, vous trouvez rapidement des éclaircissements au sujet du développement d'extensions.

La vue d'ensemble a été limitée aux répertoires contenant du code et à ceux qui présentent un certain intérêt.

```
t3lib/  
t3lib/gfx/  
t3lib/stdcdb/  
t3lib -> typo3/sysexst/cms/t3lib  
typo3/  
typo3/t3lib -> ../t3lib
```

```

typo3/install/
typo3/ext/
typo3/mod/
typo3/sysex/
typo3/sysex/cms/
typo3/sysex/cms/tslib/
typo3/sysex/lang/

```

Il y a trois répertoires de base : `t3lib/`, `tslib/` et `typo3/`.

t3lib/

Le répertoire `t3lib/` reprend les fichiers `class.t3lib_*.php` qui eux-mêmes contiennent des classes PHP. Certaines d'entre elles sont des fonctionnalités du cœur du système, mais il y a aussi des classes de base pour les modules et les services. D'autres classes reprennent des fonctions utilisées à la fois par les applications frontend et backend. La section suivante, *Bibliothèques*, fournit une vue d'ensemble.

```

t3lib/
  class.gzip_encode.php
  class.t3lib_befunc.php
  class.t3lib_div.php
  class.t3lib_extmgm.php
  class.t3lib_extobjbase.php
  class.t3lib_iconworks.php
  class.t3lib_sbase.php
  class.t3lib_sbase.php
  ...
  class.t3lib_xml.php
  config_default.php

```

Le fichier `config_default.php` comprend la configuration par défaut, que vous pouvez adapter dans le fichier `typo3conf/localconf.php` avec la variable de configuration `$TYPO3_CONF_VARS`.

```

jsfunc.evalfield.js
jsfunc.menu.js
jsfunc.updateform.js
jsfunc.validateform.js

```

Par exemple, les fichiers JavaScript contiennent des fonctions pour vérifier les entrées des TCEForms (formulaires dans le backend).

```
thumbs.php
```

`thumb.php` est utilisé dans l'attribut `src` des balises `<img>` pour générer des vignettes.

```
gfx/
```

Ce répertoire contient toutes les icônes et toutes les images utilisées par TYPO3. Vous pouvez les adapter à l'aide des habillages.

```
stddb/  
  load_ext_tables.php  
  tables.php  
  tables.sql  
  tbl_be.php
```

Un certain nombre de tables de base de données de base sont définies ici, mais ce n'est pas le cas des tables du CMS.

tslib/

Ce répertoire est un lien symbolique vers l'extension CMS. Il est décrit plus en détail dans la suite de cette section.

```
tslib -> typo3/sysextd/cms/tslib
```

typo3/

C'est dans ce répertoire que se trouve le backend, de même que l'outil d'installation et les répertoires d'extensions `ext/` et `sysext/`.

```
typo3/  
  alt_clickmenu.php  
  alt_db_navframe.php  
  alt_doc.php  
  ...  
  init.php  
  md5.js  
  stylesheet.css  
  tce_db.php  
  tce_file.php  
  template.php  
  wizard_rte.php  
  wizard_table.php  
  wizard_tsconfig.php
```

Les fichiers de ce répertoire comprennent des scripts et des classes, ainsi que des fichiers JavaScript et CSS représentant les fonctions de base du backend.

```
install/
```

C'est ici que l'on trouve l'outil d'installation qui est d'habitude appelé via l'URL du site auquel on ajoute `.../typo3/install/`.

```
mod/  
  doc/  
  file/  
  help/  
  tools/  
    em/  
  user/  
  web/
```

Certains modules backend et leurs fichiers `conf.php` sont situés dans le répertoire `mod/`. Les scripts du module eux-mêmes se trouvent souvent dans le répertoire `typo3/`. Normalement, ils devraient se trouver sous forme d'extension dans le répertoire `ext/` ; ce n'est pas le cas ici pour des raisons historiques. Les nouveaux modules sont dans tous les cas implémentés en tant qu'extensions.

```
ext/
  aboutmodules/
  ...
  wizard_sortpages/
```

Les extensions globales sont installées dans le répertoire `ext/`. En général, il s'agit de celles fournies par TYPO3, et contenant en grande partie des modules backend.

```
sysext/
```

Le répertoire `sysext/` comprend des extensions du système. Celles-ci ne peuvent normalement pas être désinstallées avec l'EM. Par contre, une mise à jour avec l'EM est possible si l'option `em_systemInstall` est activée dans `$TYPO3_CONF_VARS` (outil d'installation).

```
cms/
  ext_tables.php
  ext_tables.sql
  locallang_tca.php
  locallang_ttc.php
  tbl_cms.php
  tbl_tt_content.php
```

L'extension CMS définit des tables supplémentaires avec les fichiers ci-dessus, ainsi que de nouveaux champs et types de pages dans la table `pages`.

```
tslib/
  class.tslib_content.php
  class.tslib_fe.php
  class.tslib_fetce.php
  class.tslib_feuserauth.php
  class.tslib_gifbuilder.php
  class.tslib_menu.php
  class.tslib_pagegen.php
  class.tslib_pibase.php
  class.tslib_search.php
  index_ts.php
  pagegen.php
  publish.php
  showpic.php
```

Le répertoire `tslib/` comprend le frontend et le code de génération des pages HTML pour le site Web. Les classes de `t3lib/` sont également utilisées.

```
lang/
  lang.php
  locallang_core.ar.php
```

```
locallang_core.bg.php
locallang_core.br.php
locallang_core.ch.php
locallang_core.cz.php
locallang_core.de.php
```

L'extension **lang** du système implémente dans le fichier **lang.php** le multilinguisme du back-end. Les fichiers **locallang** contiennent chacun des traductions pour les modules standards et pour les différentes fonctions du cœur du système.

7.4.4 Bibliothèques

Référence 569519

TYPO3 fournit une série de classes PHP que vous pouvez utiliser dans vos projets. À part les bibliothèques du frontend situées dans **tslib/**, et quelques classes du backend qui se trouvent dans le répertoire **typo3/**, toutes les autres bibliothèques sont dans **t3lib/**.

Nous présentons d'abord une vue d'ensemble des classes de **t3lib/** avant de nous tourner vers les bibliothèques du frontend et du backend dans les sections suivantes.

Rappelez-vous que vous n'allez certainement pas utiliser toutes les classes de **t3lib/** dans vos projets. D'une part, parce que les classes du cœur auxquelles vous ne devriez pas accéder directement sont situées dans ce répertoire ; d'autre part, vous y trouverez des classes qui ont été déplacées, venant des modules standards. Bien que ces classes soient disponibles pour vos projets, elles ne font pas réellement partie de l'API et pourraient changer à l'avenir. En règle générale, les classes dont la documentation est fournie par l'extension **ExtDevEval** font partie de l'API officielle.

L'API de base :

t3lib_extMgm

API d'extension

Exemple : `t3lib_extMgm::extPath('myextkey')`

t3lib_DB

Abstraction de la base de données disponible sous forme d'objet via `$GLOBALS['TYPO3_DB']`

t3lib_pageSelect

Fonctions de page disponibles dans le frontend dans l'objet `$GLOBALS['TSFE']->sys_page`

Backend

t3lib_BEfunc

Une série de fonctions du backend bien utiles

Exemple : `t3lib_BEfunc::deleteClause('pages')`

t3lib_iconWorks

Génère des icônes à utiliser dans les modules du backend et dans l'arborescence des pages

t3lib_clipboard

Met en pratique le presse-papiers pour les modules du backend

Classes de base

TYPO3 fournit les classes de base suivantes pour les modules, les fonctions des sous-modules, les services et la mise en place du RTE :

t3lib_SCbase

Classe de base pour le module backend

t3lib_extobjbase

Classe de base pour les fonctions des sous-modules

t3lib_rteapi

Classe de base pour le RTE

t3lib_svbase

Classe de base pour les services

Divers

t3lib_div

Collection de fonctions générales pour BE et FE

t3lib_exec

Appelle des applications externes indépendamment de la plate-forme ; ces applications sont ensuite exécutées avec `exec()`

t3lib_cs

Convertit le texte dans un autre codage de caractères

Vue en arborescence, arborescence des pages, arborescence des répertoires

t3lib_treeView

t3lib_pageTree extends t3lib_treeView

t3lib_browseTree extends t3lib_treeView

t3lib_folderTree extends t3lib_treeView

Ces classes génèrent par exemple l'arborescence des pages dans la zone de navigation du backend. En même temps, elles servent aussi à représenter n'importe quelle table de base de données dans une arborescence ou à collecter ses données, comme par exemple dans **Outils** → **Recent changes**.

TCE et TCEForms

t3lib_TCEmain

t3lib_transferData

t3lib_loadDBGroup

t3lib_TCEforms

t3lib_TCEforms_FE extends t3lib_TCEforms

Le TCE (moteur du cœur de TYPO3) et TCEForms, qui constituent le cœur de la manipulation de données dans la base de données (TCE) et de leur traitement dans les formulaires (TCEForms), sont implémentés par les classes ci-dessus.

Identification

t3lib_userAuth

t3lib_userAuthGroup extends t3lib_userAuth

t3lib_beUserAuth extends t3lib_userAuthGroup

t3lib_tsfeBeUserAuth extends t3lib_beUserAuth

Dans les classes *Auth se trouvent les fonctions nécessaires à l'identification des utilisateurs dans le frontend et dans le backend (beUserAuth).

TypoScript

t3lib_TSTemplate

t3lib_tsparser_ext extends t3lib_TSTemplate

t3lib_tsStyleConfig extends t3lib_tsparser_ext

t3lib_TSparser

t3lib_matchCondition

Les classes ci-dessus mettent en œuvre l'analyseur syntaxique de TypoScript et les fonctions qui l'accompagnent.

Email

t3lib_htmlmail

t3lib_formmail extends t3lib_htmlmail

t3lib_dmailer extends t3lib_htmlmail

t3lib_readmail

Ces classes contiennent des fonctions pour envoyer, lire et traiter les emails (texte et HTML). Le module **Direct Mail** se base sur ces classes.

HTML

t3lib_parsehtml

t3lib_parsehtml_proc extends t3lib_parsehtml

Ces classes servent à traiter le code HTML et sont aussi utilisées en rapport avec le RTE.

Autres

t3lib_xml

Génère du XML à partir d'enregistrements

t3lib_diff

Affiche les différences entre deux textes

t3lib_syntaxhl

Génère la mise en évidence du code source, par exemple Flexform XML ou d'autres formats en fonction des besoins

t3lib_div

Nous allons présenter la classe `t3lib_div` séparément, étant donné qu'elle est constituée d'une série de fonctions très souvent utilisées par les extensions. Pour cette raison, nous vous recommandons d'étudier la documentation API de la classe. Les groupes de fonctions suivants sont présents dans la classe `t3lib_div` :

- Traitement des données GET et POST
- Fonctions de chaînes de caractères
- Fonctions de tableaux
- Traitement HTML et XML
- Fonctions de fichiers
- Fonctions de débogage
- Informations sur le système
- Fonctions TYPO3 particulières

7.4.5 L'API d'extension

Toutes les interfaces importantes pour étendre TYPO3 sont rassemblées dans l'API d'extension. L'API est disponible dans le frontend et dans le backend après l'initialisation du système, via la classe `t3lib_extMgm`.

Référence 915540

L'API d'extension fournit des fonctions pour les tâches suivantes :

- La détermination des chemins absolus et relatifs menant aux extensions
- L'extension du TCA (*Table Configuration Array*)
- L'enregistrement des modules
- L'enregistrement de TSConfig
- L'enregistrement et l'administration des services
- L'enregistrement des plugins
- L'enregistrement de TypoScript

Arrivés à ce point de l'exposé, nous n'allons pas présenter l'API dans ses moindres détails, puisque les appels à l'API lors de l'intégration d'une extension sont générés automatiquement par l'extension Kickstarter. Vous pouvez obtenir la documentation de l'API en utilisant par exemple le module **Outils** → **ExtDevEval**.

En guise d'exemple, nous introduisons brièvement les fonctions suivantes, car elles reviennent fréquemment dans les plugins et les modules. Toutes ces fonctions ont le paramètre **\$key** en commun contenant la clé d'extension qui est passée en argument pour sélectionner des données. En outre, notez que la classe **t3lib_extMgm** n'est pas instanciée, de sorte que ses fonctions peuvent être appelées directement.

Exemple :

```
$extPath = t3lib_extMgm::extPath('maCleextension');
```

function isLoaded(\$key, \$exitOnError=0)

Vérifie si une extension est installée ; les fonctions qui la suivent produisent une erreur si l'extension spécifiée n'est pas installée. Cette fonction génère aussi une erreur si vous donnez la valeur **true** au paramètre **\$exitOnError**.

function extPath(\$key, \$script='')

Cette fonction renvoie le chemin absolu de l'extension. Si nécessaire, vous pouvez spécifier un nom de fichier en argument qui sera joint au chemin dans **\$script**.

function extRelPath(\$key)

Détermine le chemin relatif de l'extension par rapport au répertoire du backend (TYPO3_mainDir, typo3/) et le renvoie.

function siteRelPath(\$key)

Fournit le chemin relatif du répertoire du site Web par rapport à l'extension.

Souvenez-vous que la détermination des chemins des extensions requiert plus de temps que l'accès aux variables. Pour cette raison, nous vous recommandons de sauver les chemins dans des variables et d'utiliser ces variables, lorsque le chemin correspondant est souvent utilisé.

7.4.6 Structure de base de données

Référence 641730

Le schéma de la base de données de TYPO3 est relativement simple. En tant que professionnel des bases de données et gourou SQL, vous aurez probablement quelques suggestions pour améliorer ce schéma. Il y a deux raisons qui expliquent la structure de la base de données de TYPO3 : d'une part, elle est basée sur une approche pragmatique, comme le reste de TYPO3; d'autre part, cette base de données a été développée lorsque MySQL ne pouvait pas encore manipuler les *subselects* ou les *unions*, par exemple. Cela signifie que TYPO3 utilise un concept dans lequel la logique est implémentée le moins possible dans la base de données, et l'est plutôt dans PHP.

Même si TYPO3 supporte les relations MM (many-to-many), des champs de texte sont utilisés de temps à autre avec des listes d'ID séparés par des virgules. Les développeurs peuvent critiquer cette pratique, mais l'utilisation des relations MM n'apporte pas dans ce cas de gros avantages du point de vue de la logique de programmation. Vous pouvez évidemment vous

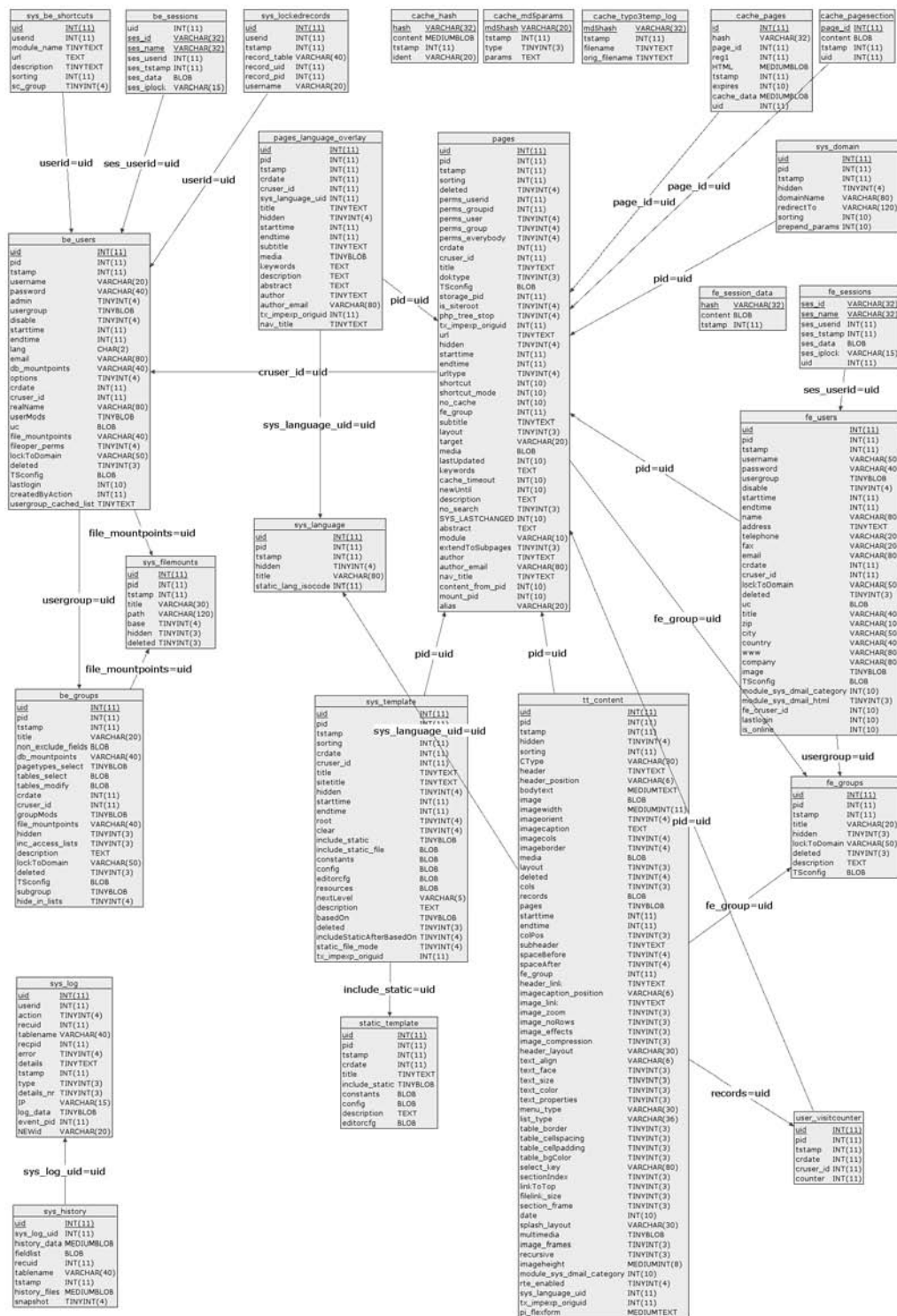


Figure 7.19:
Les tables les plus importantes de TYPO3 et de l'extension CMS ainsi que les relations principales

servir de toutes les relations nécessaires à vos applications. Mais, pour qu'elles soient supportées lors du traitement des enregistrements dans le backend, vous devrez certainement fournir du travail supplémentaire.

Arborescence des pages et éléments de contenu

L'élément central dans la structure de base de données est la page, à laquelle est assigné un ID unique, de sorte qu'elle puisse être référencée et liée par d'autres pages. Les pages reçoivent automatiquement un ID lors de leur création. Cet ID est enregistré dans le champ `uid` (*unique id*) des pages de la table de base de données. La page est affichée dans le frontend par l'URL avec l'ID *ad hoc*, par exemple `http://www.votre-domaine.com/index.php?id=18`. Ce code a pour effet d'appeler la page « B2C Accueil » dans l'application courante. Ce principe permet d'appeler de manière univoque n'importe quelle page via le frontend. L'ID assigné par le système à une page apparaît si vous placez la souris sur l'icône de la page située à côté du titre de page [1].

Dans TYPO3, tous les éléments de contenu et les enregistrements sont sauves dans une arborescence des pages comparable à l'arborescence des répertoires dans un système de fichiers. Cette arborescence est constituée de pages ; l'assignation d'une page à une page d'un niveau supérieur se réalise en utilisant le champ `pid` (parent id) dans lequel l'uid de la page parent sera enregistré.

Figure 7.20:
Arborescence des
pages [1] affichée
avec phpMyAdmin et
champs sélectionnés
de la table pages [2]



	uid	pid	sorting	title	doktype
	1472	0		Contenus	1
	6	0		448 Portail	2
	49	0		960 B2C-Accueil	1
	10	0		704 B2C-Accueil	4
	4	0		1728 Gabarit	254
	76	0		1216 B2C-Accueil	2
	173	0		2496 bidon	1
	181	0		1984 Typoscript	254
	190	0		2240 B2C-Accueil	1
	21	18		2304 Corbeille	255
	31	18		1280 Commerçants	2
	33	18		768 Produits	2
	38	18		512 Evénements	2
	43	10		256 Actualités	2
	170	18		1344 Fonctions	254
	171	18		64 Accueil	4
	102	43		256 Nouveautés-Détails	5

L'arborescence représente la structure du site. Celle-ci est le concept central autour duquel s'organisent toutes les données. Les menus et les sous-menus d'un site Web sont générés automatiquement à partir de cette structure.

Dans l'exemple d'arborescence des pages, « B2C Accueil » est la page de démarrage du site Web dans le niveau racine ; elle n'a pas de page parent. Son `pid` est donc fixé à la valeur 0. Toutes ses sous-pages du premier niveau contiennent une référence à « B2C Accueil » via la valeur 18 de leur `pid`. De même, la page « Nouveautés-Détails » du second niveau est assignée, via le `pid` 43, à la page « Actualités » dont l'uid est 43. L'ordre de l'affichage dans l'arborescence des pages et dans les menus est déterminé par le champ `sorting`.

Ceci s'applique également aux éléments de contenu de la page qui, tant qu'ils ne sont pas des extensions, sont sauvegardés dans la table de base de données `tt_content`. Lorsqu'ils sont créés, ils reçoivent eux aussi un `uid` unique ; dans le même temps, ils sont reliés—selon l'endroit où ils ont été créés—à la page correspondante, via leur `pid`. L'ordre de l'affichage est également organisé via le champ `sorting`. Pour assigner l'enregistrement à l'un des types de contenu disponibles (par exemple texte, image, formulaire, tableau, ...), utilisez le champ `CType`.

À la figure suivante, vous voyez comment les éléments de contenu de la table `tt_content` sont assignés aux pages à l'aide du champ `pid` [1]. Dans notre cas, il s'agit de la page « Accueil » avec un `uid` de 163. Dans l'affichage liste du backend, vous voyez les changements dans le champ `sorting` lorsque vous déplacez les éléments de contenu ([2] et [3]).

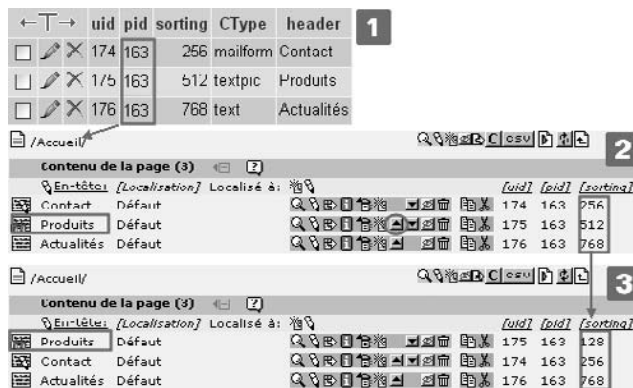


Figure 7.21:
Liens entre les
champs `uid`, `pid` et
`sorting` dans la table
de base de données
`tt_content` dans
`phpMyAdmin` [1] et
dans l'affichage liste
du backend [2]/[3]

Le rédacteur n'entre pas en contact avec cet aspect technique lorsqu'il édite les en-têtes de page ou les éléments de contenu. Il ou elle détermine les éléments de contenu assignés à chaque page avec l'interface utilisateur, assigne un type au contenu et définit l'ordre d'affichage en déterminant ou en modifiant l'ordre des éléments de contenu.

Tables et champs

Nous expliquons ici brièvement la structure de base de la table de base de données de TYPO3, même si le Kickstarter fournit déjà les champs standards nécessaires lors de la création d'une table. Ceci est particulièrement intéressant lorsque vous intégrez des applications existantes dans TYPO3.

Pour pouvoir éditer des tables dans le backend, vous devez vous référer à quelques conventions définies par le TCA (voir la prochaine section). Chaque table doit contenir les deux champs obligatoires, `uid` et `pid`. Le champ `uid` (*unique ID*) est un champ qui s'incrémente automatiquement, et dont la valeur est unique pour les enregistrements de cette table. Le champ `pid` (*page ID*) contient l'ID de la page à laquelle l'enregistrement appartient, c'est-à-dire l'`uid` dans la table `pages`.

De plus, il y a plusieurs champs standards qui ne sont pas obligatoires et dont la dénomination peut varier. Par souci de clarté, il est toutefois conseillé de conserver les noms standards. Voici quelques champs standards typiques :

Tableau 7.1:
Champs standards
dans les tables

Champ	Signification
title	est affiché comme titre dans le backend
nav_title	titre alternatif à utiliser dans les menus
crdate	date à laquelle l'enregistrement a été créé (timestamp Unix)
cruser_id	ID de l'utilisateur qui a créé l'enregistrement
tstamp	date de la dernière modification (timestamp Unix)
sorting	pour déterminer l'ordre manuellement
deleted	marque un enregistrement comme effacé
hidden	spécifie qu'un enregistrement est caché (dans le frontend)
starttime	date à partir de laquelle l'enregistrement devient actif dans le FE (timestamp Unix)
endtime	date à partir de laquelle l'enregistrement devient inactif dans le FE (timestamp Unix)

7.4.7 Base de données, TCA et TCEForms

Référence 392081

Comme dans tout système de gestion de base de données impliquant plus que les possibilités d'outils simples, l'information méta est nécessaire pour décrire la structure de la base de données et de ses champs. Sans cette information, les formulaires de saisie pour les enregistrements dans le backend n'auraient pas pu être développés.

phpMyAdmin permet aussi, exactement comme TYPO3, d'éditer des ensembles de données dans une table de base de données à l'aide d'une interface Web. Alors, quelle est la différence entre les deux ? En fait, phpMyAdmin génère des formulaires de saisie pour les enregistrements à partir de la définition de la base de données elle-même. Par conséquent, phpMyAdmin est conscient du fait qu'un champ donné ne peut accepter que des valeurs entières, puisque ce champ est défini comme tel dans la base de données. En revanche, il ne sait pas si une donnée n'accepte des valeurs que dans une certaine fourchette, et il n'est donc pas en mesure d'afficher un champ de sélection avec des valeurs prédéfinies. Des paramètres supplémentaires sont nécessaires pour exécuter ce genre de tâche, qui n'est pas sous-entendue dans la définition de la base de données. TYPO3 se sert de ce type d'information pour traiter correctement les tables de base de données, les enregistrements et les champs, ainsi que pour rendre disponibles des fonctionnalités que phpMyAdmin ne peut fournir.

Figure 7.22:
Traitement des
données dans
phpMyAdmin et
TYPO3 (TCEForms)

hidden	tinyint(4) unsigned	☐	0
starttime	int(11) unsigned	☐	0
endtime	int(11) unsigned	☐	0
fe_group	int(11)	☐	1

Options générales
Cacher: ☐ [?] Lancement: ☐ [?] Arrêt: ☐ [?] Accès: ☐ [?]

À l'aide de cette information méta (TCA) et des TCEForms, l'édition des formulaires est rendue disponible dans le backend. La donnée saisie est vérifiée par le TYPO3 Core Engine (TCE, c'est-à-dire l'endroit où les définitions TCA sont utilisées) et écrite dans la base de données.

L'information méta nécessaire à TYPO3 est disponible dans le TCA (*Table Configuration Array*) dans la variable globale `$TCA`. La donnée TCA est définie dans les fichiers suivants, qui peuvent servir de source d'information pour la programmation des extensions :

```
t3lib/stdtdb/tables.php
t3lib/stdtdb/tbl_be.php
typo3/sysext/cms/ext_tables.php
typo3/sysext/cms/tbl cms.php
typo3/sysext/cms/tbl_tt_content.php
.../extension/ext_tables.php
```

La définition actuelle du tableau TCA peut être représentée par un arbre dans lequel vous accédez à des valeurs, ce qui est par exemple très utile lorsque vous voulez localiser une erreur.

Ci-après, nous présentons une vue d'ensemble des options du TCA. Une liste exhaustive des options sortirait du cadre de ce livre. Une telle liste se trouve à la référence mentionnée ci-contre. Dans la plupart des cas, les options de définition de champs du Kickstarter sont suffisantes, de sorte que vous n'aurez que rarement besoin de consulter cette référence.

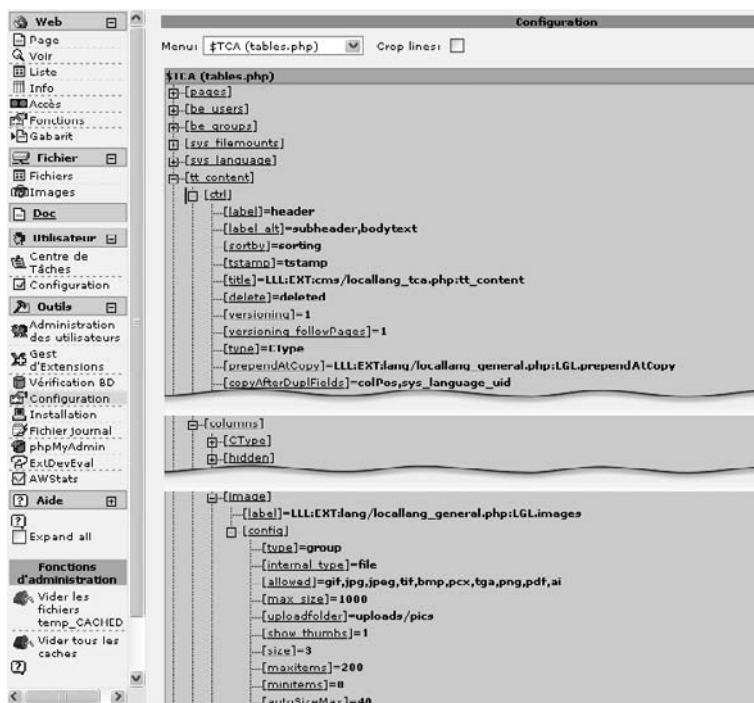


Figure 7.23:
Navigateur TCA
contenant des
extraits de la
définition tt_content

Voici un extrait du tableau TCA qui définit la table `tt_content`. Comme nous le savons, cette table sert à sauvegarder des éléments de contenu (du texte, des images, ...).

La définition TCA démarre par le tableau `ctrl` qui contient quelques détails de données méta pour l'utilisation des tables. De plus, on définit le champ repris dans l'affichage liste en mode restreint.

```
$TCA['tt_content'] = Array (
    'ctrl' => Array (
        'label' => 'header',
        'label_alt' => 'subheader,bodytext',
```

Les trois lignes suivantes déterminent le champ qui servira au tri (**sortby**), celui qui contiendra la date de dernière modification (**tstamp**) et la manière de nommer la table dans le backend (**title**). A cette fin, on crée une référence vers le fichier externe (**locallang**) qui contient les valeurs des titres des champs dans différentes langues.

```
        'sortby' => 'sorting',
        'tstamp' => 'tstamp',
        'title' => 'LLL:EXT:cms/locallang_tca.php:tt_content',

        'delete' => 'deleted',
        'type' => 'CType',
        'prependAtCopy' => 'LLL:EXT:lang/locallang_general.php:LGL.prependAtCopy',
        'copyAfterDuplFields' => 'colPos,sys_language_uid',
        'useColumnsForDefaultValues' => 'colPos,sys_language_uid',
```

Le tableau suivant, **enablecolumns**, définit les champs qui déterminent si un enregistrement sera visible ou non dans le frontend. La fonction **enableFields()** génère du code SQL sur cette base, code qui filtre les enregistrements invisibles.

```
        'enablecolumns' => Array (
            'disabled' => 'hidden',
            'starttime' => 'starttime',
            'endtime' => 'endtime',
            'fe_group' => 'fe_group',
        ),
        ...
        'mainpalette' => '1',
        'thumbnail' => 'image',
    )
    'interface' => Array (
        'always_description' => 0,
        'showRecordFieldList' => 'CType,header,header_link,bodytext,image,...'
    ),
```

Dans le tableau **columns** sont définis les champs des tables de base de données. Ils comprennent essentiellement de l'information sur le type de champ de formulaire utilisé pour éditer les champs de la table (champ de sélection, case à cocher, ...) ainsi que leur type (texte, entier, ...). En outre, vous pouvez définir des fourchettes de valeurs ou fixer des valeurs standards.

Les TCEForms se servent de cette information pour générer les champs de formulaire correspondants. Le TCA forme avec ceux-ci la base de la manipulation des enregistrements (trier, cacher, ...) d'une part, et la base de leur édition avec TCEForms d'autre part.

```
'columns' => Array (
```

Le champ `CType` est ici défini comme un champ de sélection. Il spécifie le type de contenu (texte, image, ...).

```
'CType' => Array (
  'label' => 'LLL:EXT:lang/locallang_general.php:LGL.type',
  'config' => Array (
    'type' => 'select',
    'items' => Array (
      Array('LLL:EXT:cms/locallang_ttc.php:CType.I.0', 'header'),
      Array('LLL:EXT:cms/locallang_ttc.php:CType.I.1', 'text'),
      Array('LLL:EXT:cms/locallang_ttc.php:CType.I.2', 'textpic'),
      Array('LLL:EXT:cms/locallang_ttc.php:CType.I.3', 'image'),
      Array('LLL:EXT:cms/locallang_ttc.php:CType.I.4', 'bullets'),
      ...
    ),
    'default' => 'text'
  )
),
```

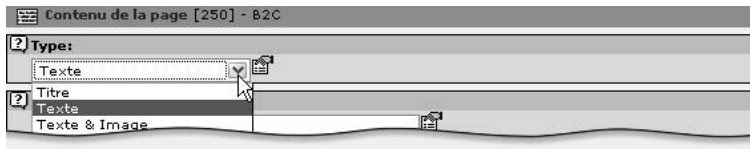


Figure 7.24:
TCEForms affiche un
champ de sélection
pour le champ `CType`

Le champ `hidden` est défini comme une case à cocher ; il sert à désactiver l'enregistrement dans le frontend (cacher).

```
'hidden' => Array (
  'exclude' => 1,
  'label' => 'LLL:EXT:lang/locallang_general.php:LGL.hidden',
  'config' => Array (
    'type' => 'check'
  )
),
...
```

`bodytext` est la zone de saisie qui récupère le texte des éléments de contenu **Text** et **Text with image** par exemple.

```
'bodytext' => Array (
  'label' => 'LLL:EXT:lang/locallang_general.php:LGL.text',
  'config' => Array (
    'type' => 'text',
    'cols' => '48',
    'rows' => '5',
  )
),
...
```


La définition d'**image** est un peu plus complexe. Il s'agit d'un champ de sélection de type **group** qui contient certains détails supplémentaires. Par exemple, le nombre maximal d'entrées est limité à 200 par **maxitems**.

```
'image' => Array (
  'label' => 'LLL:EXT:lang/locallang_general.php:LGL.images',
  'config' => Array (
    'type' => 'group',
    'internal_type' => 'file',
    'allowed' => $GLOBALS['TYPO3_CONF_VARS']['GFX']['imagefile_ext'],
    'max_size' => '1000',
    'uploadfolder' => 'uploads/pics',
    'show_thumbs' => '1',
    'size' => '3',
    'maxitems' => '200',
    'minitems' => '0',
    'autoSizeMax' => 40,
  )
),
...
),
```

Figure 7.25:
Champ de sélection
du type **group** pour le
champ **image**



Dans le tableau **types** sont définis les différents types de contenu (texte, image, ...). Par exemple, les champs sont affichés en fonction du type d'enregistrement. C'est la valeur du champ **CType** qui détermine le type d'un enregistrement donné, car celui-ci a été entré dans le tableau **ctrl** comme tel, via la clé **type**. Si cette valeur est **text**, c'est-à-dire que l'enregistrement doit être de type **text**, le champ **CType** est affiché avec la palette 4, le champ **header** avec la palette 3, le champ **bodytext** avec la palette 9, etc.

```
'types' => Array (
  '1' => Array('showitem' => 'CType'),
  'header' => Array('showitem' =>
    'CType;;4;button;1-1-1, header;;3;;2-2-2, subheader;;8'),
  'text' => Array('showitem' =>
    'CType;;4;button;1-1-1, header;;3;;2-2-2, bodytext;;9;...
  ...
),
```

Les palettes sont définies dans le tableau **palettes**. La définition ci-dessus, **header;;3;;** spécifie que la palette 3 doit être utilisée pour le champ **header**. Cette palette affiche les champs concernant l'en-tête et le champ **date**, comme vous le voyez ci-dessous.

```
'palettes' => Array (
  '1' => Array('showitem' => 'hidden, starttime, endtime, fe_group'),
  '2' => Array('showitem' => 'imagecols, image_noRows, imageborder'),
```

```

'3' => Array('showitem' =>
    'header_position, header_layout, header_link, date'),
...
)
);

```

Les champs eux-mêmes sont déjà définis dans le tableau `columns` du dessus et sont présentés comme une palette en une seule ligne, au lieu de plusieurs lignes.



Figure 7.26:
Palette 3 sous le
champ header

7.4.8 Flexforms

Les champs de base de données sont dans tous les cas utilisés d'une seule manière dans la définition du TCA. Cela signifie qu'un champ dans lequel on saisit un nombre ne peut être utilisé pour enregistrer une date, bien que ce soit dans les deux cas une valeur entière qui est sauvegardée. De plus, une table de base de données est une structure figée. Les Flexforms offrent une méthode d'enregistrement des données plus flexible, comme l'autorise le format XML.

Référence 353987

Les Flexforms permettent d'effectuer plusieurs types de tâches. Nous les connaissons déjà via **TemplaVoilà**. Vous trouverez de l'aide en consultant la référence ci-contre.

Puisque aucune option de requête n'existe actuellement pour les données Flexform, leur utilité se limite aux tâches dans lesquelles les requêtes SQL ne sont pas requises pour rechercher des données enregistrées dans les Flexforms : par exemple, les options pour configurer les plugins via l'élément de contenu **Insert Plugin**.

Une manière de rendre des options disponibles pour les plugins est d'ajouter un ou plusieurs champs à la table `tt_content` afin de sauvegarder la configuration. La définition TCA correspondante affiche dès lors les options à l'aide de l'élément de contenu **Insert Plugin**. Le désavantage de cette méthode est que chaque plugin ajoute ses propres champs dans `tt_content`, ce qui ralentit inutilement les requêtes de la base de données.

Une possibilité plus élégante consiste à utiliser les Flexforms qui n'ont besoin que d'un seul champ. En fait, `tt_content` contient déjà le champ `pi_flexform` qui, comme son nom l'indique, est justement prévu pour cette méthode.²

Ci-dessous, nous vous indiquons comment utiliser les Flexforms avec le champ `pi_flexform` en prenant comme exemple l'extension **Newloginbox**.

²Robert Lemke, codéveloppeur de TemplaVoilà et des Flexforms, insiste sur le fait que l'utilisation des Flexforms pour configurer les plugins est recommandée : « Anyone who still inserts their own configuration fields into `tt_content` will be tarred and feathered. » Les auteurs approuvent cette recommandation d'un point de vue technique, mais émettent des doutes sur le type de punition préconisée. Au lieu de cela, nous menaçons de recourir à la moquerie ou à toute autre méthode pouvant aboutir à un dommage social lors du prochain Snowboard Tour de TYPO3.

Figure 7.27:
Sauvegarde des
paramètres de
configuration de
newloginbox avec des
Flexforms

Afin de configurer un plugin comme `newloginbox` via des Flexforms, les conditions suivantes doivent être satisfaites :

- La définition TCA du champ `tt_content.pi_flexform` est de type `flex`
- Activation du champ `pi_flexform` pour le plugin dans le TCA
- Création d'une définition de structure de données
- Activation de la structure de données pour le plugin
- Lecture de la configuration à partir des données Flexform dans le plugin

Le champ `pi_flexform` est déjà prédéfini comme un champ de type `flex`. La définition dans le TCA se présente comme suit :

```
'pi_flexform' => array(
    'label' => 'LLL:EXT:cms/locallang_ttc.php:pi_flexform',
    'config' => Array (
        'type' => 'flex',
        'ds_pointerField' => 'list_type',
        'ds' => array(
            'default' => '
                <T3DataStructure>
                <ROOT>
                <type>array</type>
                <el>
                    <xmlTitle>
                        <TCEforms>
                            <label>The Title:</label>
                            <config>
                                <type>input</type>
                                <size>48</size>
                            </config>
                        </TCEforms>
                    </xmlTitle>
                </el>
                </ROOT>
            </T3DataStructure>
        '
    )
),
```

Comme vous le voyez, le tableau `config` contient les clés `ds` (*Data Structure*) et `ds_pointerField`. À son tour, `ds` contient un tableau dont les éléments contiennent eux-mêmes les structures de données Flexform. Ces structures de données sont définies en XML et comprennent de l'information sur les champs désirés et leurs types de données.

Dans le fichier `ext_tables.php` de l'extension `newloginbox`, nous ajoutons le champ `pi_flexform` au plugin pour l'affichage :

```
$TCA['tt_content']['types']['list']['subtypes_addlist'][$_EXTKEY.'_pi1']
='pi_flexform';
```

Avec cette configuration, il n'y a toujours aucun élément Flexform visible dans l'élément de contenu **Insert Plugin**. De fait, aucune structure de données n'a encore été enregistrée avec la clé `newloginbox_pi1` dans le tableau `ds`. Le champ `list_type` (dans le BE sous la forme d'un champ de sélection avec les plugins disponibles) dans la table `tt_content` de cet exemple contient la valeur `newloginbox_pi1`, car ce plugin a été choisi dans l'élément de contenu **Insert Plugin**.

Le paramètre `ds_pointerField` spécifie que la valeur du champ `list_type` est utilisée en tant que clé du tableau `ds`. Par conséquent, `list_type` détermine la structure du fichier.

Expliquer les détails de la structure de données `newloginbox` serait trop complexe dans ce cadre, et nous vous renvoyons donc à la référence. La structure de données est configurée de la même manière que la structure de l'exemple précédent, via la clé `default`. La partie intéressante est celle comprise entre les balises `<TCEForms>`. Ces définitions sont les mêmes que dans le tableau `TCA`—sauf qu'elles se présentent sous la forme d'une structure XML.

La structure pour le plugin se trouve dans l'extension, dans le fichier `flexform_ds.xml`. Elle est enregistrée via l'API d'extension dans la définition `TCA` du fichier `ext_tables.php`.

```
t3lib_extMgm::addPiFlexFormValue($_EXTKEY.'_pi1',
'FILE:EXT:newloginbox/flexform_ds.xml');
```

L'élément Flexform est à présent affiché dans le formulaire plugin. La donnée saisie dans ce formulaire est transférée via le TCE vers du XML et est sauvée dans le champ `pi_flexform`. La classe de base pour les plugins, `tslib_pibase`, contient des méthodes servant à la manipulation des données dans le plugin.

Le champ Flexform `pi_flexform` doit d'abord être initialisé, ce que vous devriez faire au début du code du plugin :

```
// Init FlexForm configuration for plugin:
$this->pi_initPiFlexForm();
```

Vous accédez au champ `pi_flexform` à l'intérieur du plugin, avec `$this->cObj->data['pi_flexform']`. Puisque ce champ contient un tableau dans un format précis après initialisation, vous y accédez via la méthode `pi_getFFvalue()` qui renvoie les données des pseudo-champs qu'il contient.

```
$this->pi_getFFvalue($this->cObj->data['pi_flexform'],
'show_forgot_password','sDEF')
```

Le plugin `newloginbox` contient plusieurs options classées selon les onglets de l'index (**General**, **Welcome**, ...). Les données sont organisées en ce qu'on appelle des « feuilles ». La ligne de code ci-dessus fournit la valeur du pseudo-champ `show_forgot_password` à partir de la feuille `sDEF`.

Il y a beaucoup d'endroits où les Flexforms peuvent s'avérer utiles. Vous pouvez bien entendu les utiliser dans vos tables de bases de données afin, par exemple, de sauver les changements dans les produits d'une boutique en ligne. En outre, les Flexforms possèdent un framework permettant d'éditer les données XML, ce qui est certainement d'un grand intérêt dans de nombreuses applications.

7.4.9 TYPO3 Core Engine (TCE)

Référence 119361

Le *TYPO3 Core Engine* (TCE) est situé dans la classe `t3lib_TCEmain`. Il est responsable de la manipulation des données enregistrées dans les tables de bases de données définies par le TCA. Le TCE garantit la cohérence lors du traitement des enregistrements ; il est utilisé par les modules backend. De plus, il est possible d'accéder au TCE via les scripts `typo3/tce_db.php` et `typo3/tce_file.php` en spécifiant des commandes dans ces scripts. Nous montrerons cela plus tard à l'aide d'un exemple.

Le TCE utilise les données méta de la définition du TCA pour vérifier si les données saisies sont valides. Les permissions d'accès sont aussi prises en compte et les identifications sont historisées. Le TCE est conçu pour fonctionner de pair avec les TCEForms. Cela signifie que le TCE connaît les formats de données des TCEForms et en tient compte. Si, par exemple, le champ est défini comme une relation MM dans le TCA, vous devez d'abord rechercher les enregistrements de la relation avant que le formulaire d'édition ne soit affiché, de sorte que les enregistrements puissent être sélectionnés par ce formulaire. C'est exactement de la même manière que le TCE doit configurer les enregistrements sélectionnés comme des relations MM dans la base de données.

Si le TCE est utilisé directement via la classe `t3lib_TCEmain`, il apparaîtra comme suit :

```
require_once (PATH_t3lib.'class.t3lib_tcemain.php');
$tce = t3lib_div::makeInstance('t3lib_TCEmain');
$tce->start($data,$cmd);
$tce->process_datamap();
$tce->process_cmdmap();
```

Le TCE est contrôlé par deux tableaux, dont l'un peut contenir des données, `datamap`, et l'autre des commandes, `cmdmap`. Ces deux tableaux sont inclus dans le TCE par la commande `$tce->start($data,$cmd)`. Les tableaux correspondants sont traités avec `process_datamap()` et `process_cmdmap()`.

Commandes (cmdmap)

Les commandes suivantes peuvent être appliquées aux enregistrements (ou aux pages) via le `cmdmap` :

`delete`

Supprime un enregistrement.

copy

Copie un enregistrement.

move

Déplace un enregistrement.

Le tableau `cmdmap` a le format suivant : `$cmd[nom de la table][uid][commande] = valeur`

nom de la table

Nom de la table de la base de données à traiter ; elle doit être définie dans le TCA

uid

uid de l'enregistrement

commande

Les commandes à exécuter : `delete`, `copy` ou `move`

valeur

Pour la commande `delete`, `value` reçoit la valeur `TRUE`. Pour les commandes `copy` et `move`, `value` devrait recevoir un `page id`. Cette valeur peut être négative, ce qui signifie alors qu'un enregistrement est inséré après la page ayant ce `page id` (en valeur absolue), au lieu d'être inséré dans la page si la valeur avait été positive.

Les commandes sont donc créées comme suit :

```
$cmd[nom de la table][uid]['delete'] = TRUE
```

```
$cmd[nom de la table][uid]['copy'] = +/-pid
```

```
$cmd[nom de la table][uid]['move'] = +/-pid
```

L'exemple suivant supprime l'enregistrement avec l'`uid=54` de la table `tt_content` :

```
$cmd = array();
$cmd['tt_content'][54]['delete'] = 1;
$tce->start(array(), $cmd);
$tce->process_cmdmap();
```

Cette commande déplace la page avec l'`uid=1203` en première position à l'intérieur de la page qui a l'`uid=303`.

```
$cmd = array();
$cmd['pages'][1203]['move'] = 303;
$tce->start(array(), $cmd);
$tce->process_cmdmap();
```

Dans le prochain exemple, la page `1203` n'est pas copiée dans la page `303`, mais au même niveau, directement après la page (`303`).

```
$cmd = array();
$cmd['pages'][1203]['copy'] = -303;
$tce->start(array(), $cmd);
$tce->process_cmdmap();
```

Vous pouvez spécifier plusieurs options dans les commandes. Par exemple, l'option `copyTree` fixe la profondeur (le nombre de niveaux) des pages copiées.

```
$cmd = array();  
$cmd['pages'][1203]['copy'] = 303;  
$tce->copyTree = 3;  
$tce->start(array(), $cmd);  
$tce->process_cmdmap();
```

La profondeur est ici fixée à trois niveaux de sorte que la page **1203** est copiée, de même que ses sous-pages sur les trois premiers niveaux. Pour d'autres options, consultez la documentation à la référence du début de cette section.

Données (datamap)

Le tableau **datamap** sert à manipuler ou à créer des enregistrements. Son format est le suivant :

\$cmd[nom de la table][uid][nom du champ] = valeur

nom de la table

Nom de la table de la base de données à traiter ; la table doit être définie dans le TCA.

uid

L'uid de l'enregistrement à modifier ; si on crée un nouvel enregistrement, l'uid doit contenir une certaine valeur avec le préfixe **NEW**.

nom du champ

Nom du champ de la base de données à modifier ; ce champ doit être défini dans le TCA.

valeur

Nouvelle valeur du champ.

L'exemple suivant illustre le fait que plusieurs enregistrements peuvent être édités ou créés en même temps :

```
$data['pages']['NEW9823be87'] = array(  
    'title' => 'Page 1',  
    'pid' => '45'  
)
```

Il faut d'abord créer une nouvelle page avec le titre "Page 1" dans la page pour laquelle uid=45.

```
$data['pages']['NEWbe68s587'] = array(  
    'title' => 'Page 2',  
    'pid' => '-NEW9823be87'  
)
```

Une deuxième page est à présent créée après la nouvelle page, et sur le même niveau, avec le pseudo-uid **NEW9823be87**. Le concept d'uid négatif est utilisé de la même manière que pour **cmdmap**.

```
$data['pages'][9834] = array(  
    'title' => 'Nouveau titre pour cette page',  
    'subtitle' => 'Nouveau sous-titre'  
)  
$tce->start($data, array());  
$tce->process_datamap();
```

Pour terminer, nous changeons le titre de la page dont l'uid est **9834**.

Clear cache

Le TCE fournit également des fonctions pour manipuler le cache.

```
$tce->clear_cacheCmd($cacheCmd);
```

Les commandes suivantes peuvent être entrées dans `$cacheCmd` :

page id

supprime le cache pour la page spécifiée

'pages'

supprime le cache pour toutes les pages

'all'

supprime tous les caches

tce_db.php

Les fonctions fournies par le TCE peuvent aussi être appelées par le script `typo3/tce_db.php`. Les tableaux `datamap` et `cmdmap` sont passés en paramètres via GET ou POST avec `data[]` et `cmd[]`. Un exemple typique de l'utilisation de `tce_db.php` se trouve à la section 7.8.2 : fonctions de sous-module **Web** → **Fonctions** → **Assistants**.

7.4.10 SQL et tables définies dans le TCA

Alors que la manipulation des données dans la base de données est exécutée via le TCE pour garantir la cohérence des données, les requêtes de base de données sont réalisées directement par SQL. Cependant, un certain nombre d'éléments sont à prendre en considération ici. D'une part, les permissions d'accès doivent être prises en compte et d'autre part, vous devez vous assurer que les enregistrements sont actifs (cachés, supprimés, ...).

Dans les modules backend, la variable `$this->perms_clause` est précédée d'une condition SQL-WHERE qui, appliquée à la table `pages`, sélectionne uniquement les pages auxquelles l'utilisateur backend actuel a accès.

Il faut aussi voir si un enregistrement est actif ou non. Dans le backend, cela revient à vérifier si l'enregistrement est supprimé ou non. Si vous définissez un champ qui marque l'enregistrement comme supprimé, comme dans le cas de la table `tt_content`, alors ce champ doit être pris en considération pour que de tels enregistrements ne soient pas affichés dans le backend.

```
$TCA['tt_content']['ctrl']['delete'] => 'deleted'
```

Une instruction **WHERE** est créée pour le backend avec la fonction suivante :

```
$deleteClause = t3lib_BEfunc::deleteClause('table_name');
```

Il existe une méthode similaire pour le frontend dans l'objet TSFE :

```
$deleteClause = $TSFE->sys_page->deleteClause('table_name');
```

De plus, il existe une série de champs qui définissent la validité de l'enregistrement pour le frontend. Ces champs sont utilisés pour cacher l'enregistrement, pour le laisser apparaître un

certain temps, ou pour définir les permissions d'accès pour les groupes d'utilisateurs frontend. Tous ces champs sont appelés **enable fields**. Vous pouvez créer une instruction **WHERE** correspondante dans les méthodes suivantes. Dans ce cas, le code `deleteClause()` ne doit plus être repris, car la clause tenant compte des enregistrements supprimés est déjà incluse.

```
$enableFields = $TSFE->sys_page->enableFields('table_name');
```

```
$enableFields = $this->cObj->enableFields('table_name');
```

À l'intérieur des plugins, c'est cette dernière méthode qui est utilisée, de sorte qu'une requête ressemble à peu près à ceci :

```
$res = $GLOBALS['TYPO3_DB']->exec_SELECTquery(
    '*',
    $table,
    'pid IN ( '.$pidList.' )'.
    $this->cObj->enableFields($table),
    '',
    'crdate DESC');

while($row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res)) {
    ...
}
```

Référence 215109

La requête de base de données est réalisée par les méthodes de l'objet `$GLOBALS['TYPO3_DB']`. TYPO3 implémente une couche d'abstraction de base de données (DBAL pour *database abstraction layer*) via cette interface. L'API se trouve dans la classe `t3lib_DB`. Une description du DBAL est disponible à la référence ci-contre.

Si `enableFields()` ou `deleteClause()` sont utilisés correctement dans les requêtes SQL, vous ne devez pas vous soucier de savoir quels champs et quelles fonctions sont définis pour la table dans le TCA. Si les champs `starttime` et `endtime` sont ajoutés à la table par la suite (via les entrées correspondantes dans le TCA), ils seront inclus dans les requêtes sans devoir changer le code PHP du plugin, pour autant qu'un plugin utilise `enableFields()` pour les requêtes de base de données.

7.4.11 Utilisateurs, sessions et identification

Référence 358205

TYPO3 tient compte des permissions sur les pages, des enregistrements et des fichiers selon le profil d'utilisateur. Un utilisateur qui veut accéder à une ressource doit bien sûr être connu. Ceci implique deux aspects essentiels : d'une part, il faut pouvoir authentifier un utilisateur, c'est-à-dire reconnaître l'identité d'un utilisateur donné. Ceci est généralement réalisé à l'aide d'un formulaire d'identification, en vérifiant le nom d'utilisateur et le mot de passe. D'autre part, il faut pouvoir identifier continuellement un utilisateur : puisque HTTP est un protocole sans état, à chaque page appelée, il faut pouvoir déterminer si l'utilisateur est déjà connu et si les données en question sont réservées à l'utilisateur. Cette surveillance de l'utilisateur est appelée une *session*.

Un exemple typique illustrant les différences entre ces deux fonctions est celui de l'Online Shop. Alors que le panier d'achat virtuel est une fonction de la session, l'utilisateur doit s'authentifier au plus tard lors du traitement de la commande (dépendant du magasin) : cela permet de l'identifier.

Dans le contexte des applications Web, nous définissons trois types d'utilisateurs :

- les utilisateurs non-identifiés
- les utilisateurs identifiés (session utilisateur sans authentification)
- les utilisateurs authentifiés

Tandis que pour le backend, il est essentiel que l'utilisateur soit authentifié, les trois types d'utilisateurs peuvent entrer en ligne de compte dans le frontend. La plupart des sites Web sont accessibles sans identification, et ne dépendent pas non plus des sessions utilisateurs. Cela signifie qu'ils sont complètement fonctionnels pour des utilisateurs non-identifiés.

Toutefois, si des données relatives à l'utilisateur sont collectées lors d'une visite du site Web (comme pour les paniers d'achat virtuels), il faut configurer une session. Dans une session, un utilisateur est suivi lors de sa visite dans le site Web. Ce statut n'est pas le même que celui de l'authentification, puisque l'identité de l'utilisateur ne doit pas être connue dans ce cas-ci.

L'accès à des ressources précises (pages ou éléments de contenu) est contrôlé via les groupes d'utilisateurs. Pour ce faire, l'utilisateur doit être authentifié. Sans session utilisateur, l'authentification ne peut être appliquée de manière sensée.

Une session utilisateur reconnue au moyen d'un cookie a été créée automatiquement par TYPO3. Ce cookie contient seulement un ID, et donc, ne contient pas de données sur l'utilisateur. Celles-ci sont stockées dans la base de données et sont lues après identification du cookie ; elles contiennent également des données pour la session. Ces données sur l'utilisateur sont aussi utilisées pour sauver le statut de l'authentification, ce qui explique qu'un utilisateur donné ne doive s'authentifier qu'une seule fois par session.

Voici un résumé de l'identification d'un utilisateur frontend :

- Un ID d'utilisateur existant est détecté au moyen d'un cookie.
- Si l'ID n'existe pas, l'authentification est validée en vérifiant le nom d'utilisateur et le mot de passe fournis.
- Un cookie contenant l'ID d'utilisateur est alors créé.
- Si l'utilisateur est authentifié, les données sur l'utilisateur sont lues.
- Les données de la session sont lues ; par exemple, dans le cas d'un magasin, elles pourraient contenir les entrées d'un panier d'achat virtuel.

L'utilisation de cookies dans le frontend peut être désactivée par la configuration suivante :

```
$TYPO3_CONF_VARS['FE']['dontSetCookie'] = 1;
```

Afin de pouvoir toujours authentifier des utilisateurs et utiliser des sessions, vous pouvez configurer l'option `config.ftu` (*Frontend Track User*) dans le **setup** TypoScript qui permet d'identifier un utilisateur au moyen des paramètres fournis. Mais gardez à l'esprit le fait que les plugins qui ne génèrent pas de liens en utilisant correctement les fonctions du système peuvent terminer une session ; c'est pourquoi l'utilisation de cookies est recommandée.

La gestion des sessions, l'authentification et le fait de fournir des données utilisateurs sont des actions contrôlées par les classes `t3lib_*userAuth` et `tslib_feUserAuth`. Après l'identification,

et l'authentification si nécessaire, un objet de ces classes est mis à votre disposition. Cet objet garde les données concernant l'utilisateur dans le tableau `->user[]`. Il contient des méthodes pour accéder aux données de la session ou pour vérifier les permissions d'utilisateur.

Les objets d'utilisateur dans le frontend et le backend sont très similaires dans leur structure de base, étant donné que c'est la même classe de base qui est utilisée pour les deux. Mais il y a une différence entre eux, particulièrement dans les méthodes. Par exemple, la méthode `->isAdmin()` est disponible dans le backend, mais pas dans le frontend, puisqu'il n'y a pas d'utilisateur « administrateur » dans ce cas.

Les objets d'utilisateur sont disponibles dans les variables globales suivantes :

```
$GLOBALS['TSFE']->fe_user  
    Objet d'utilisateur frontend  
$GLOBALS['BE_USER']  
    Objet d'utilisateur backend
```

Voici quelques exemples de code impliquant les objets d'utilisateur.

Frontend

```
global $TSFE;  
$mySessionData = $TSFE->fe_user->getKey('ses', $mySessionKey);  
$mySessionData = ...  
$TSFE->fe_user->setKey('ses', $mySessionKey, $mySessionData);  
$TSFE->fe_user->storeSessionData();
```

Cet exemple montre comment les données de la session sont sauvegardées dans les plugins. Les données peuvent être sous n'importe quel format puisqu'elles sont enregistrées via la fonction (`serialize()`) et restaurées de manière correspondante par le système. Les données sont identifiées via une clé. Il est de bonne pratique d'utiliser dans ce but le nom du plugin, à savoir ici `$this->$prefixId`.

```
$userID = $TSFE->fe_user->user['uid'];  
$username = $TSFE->fe_user->user['username'];
```

Si un utilisateur est authentifié dans le frontend, alors ses données sont disponibles dans le tableau `user[]`.

```
$userTSconf = $TSFE->fe_user->getUserTSconf();
```

Les données TSConfig sont disponibles via `getUserTSconf()`.

Backend

```
global $BE_USER;  
$userID = $BE_USER->user['uid'];
```

Le tableau `user[]` est disponible dans l'objet d'utilisateur qui contient les données concernant l'utilisateur.

```
$userIsAdmin = $BE_USER->isAdmin();
```

Vérifiez si l'utilisateur a le statut d'administrateur avec `isAdmin()`.

```
$uc_titleLen = $BE_USER->uc['titleLen'];
```

La configuration de l'utilisateur est sauvegardée dans le tableau `uc[]` (*User Config*). Vous y trouverez par exemple des données issues du module **Utilisateur** → **configuration** mais aussi des paramètres de modules, qui sont automatiquement sauves dans ce tableau par la classe de base `t3lib_SCbase` et restaurés au prochain appel de ce module (voir la section *Framework des modules*). Les exemples ci-dessus ne présentent qu'une partie de l'API Utilisateur. Vous avez de l'information supplémentaire à votre disposition sur l'implémentation des objets d'utilisateur à la référence de cette section.

7.4.12 Programmation TYPO3 et plate-forme

En théorie, TYPO3 fonctionne sur toutes les plate-formes sur lesquelles PHP fonctionne. Il y a toutefois certaines différences selon la plate-forme, la version PHP et la configuration du serveur utilisées. Plusieurs fonctions PHP manquent ou se comportent différemment jusqu'à un certain point, surtout sous Windows.

Référence 219501

TYPO3 fournit dans son API des fonctions qui aplanissent la plupart de ces différences. Si, en outre, vous suivez certaines conventions, vos extensions devraient fonctionner sans problème dans différents environnements.

Variables du serveur et variables d'environnement

Si vous voulez accéder aux variables du serveur, n'utilisez jamais `getenv()` ou même `$HTTP_SERVER_VARS`, utilisez plutôt la fonction `t3lib_div::getIndpEnv()`. Cette fonction réunit les variables connues du serveur et elle fournit de l'information supplémentaire.

Certaines de ces variables ont trait à l'adresse URL de la page courante. L'URL se décompose comme suit :

```
[protocole]://[hôte][:[port]][chemin][?[requête]]
```

L'URL suivante est utilisée en guise d'exemple :

```
http://www.example.net:80/t3livre/index.php?id=26&tx_myext[myparam]=1234
```

L'installation TYPO3 se trouve dans le répertoire `/var/www/t3livre/` alors que le répertoire de base du serveur Web est `/var/www/`.

Information générale

REMOTE_ADDR

Adresse IP du poste client (navigateur).

REMOTE_HOST

Nom d'hôte du poste client (navigateur).

HTTP_USER_AGENT

Contient le type et le nom du client appelant (navigateur).

Exemple : User-Agent: Mozilla/5.0 (X11; U; Linux i686; de-AT; rv:1.6) Gecko/20040411
Debian/1.6-5

HTTP_ACCEPT_LANGUAGE

Les langues acceptées ou demandées par le client appelant (navigateur).

Exemple : de-de,en-gb

HTTP_HOST

[hôte][:[port]] – l'hôte requis par le client.

Exemple : www.example.net:80

TYPO3_HOST_ONLY

[hôte]

Exemple : www.example.net

TYPO3_PORT

[port]

Exemple : 80

HTTP_REFERER

[protocole]://[hôte][:[port]][chemin][?[requête]]

L'URL par laquelle la page courante a été appelée.

URL

SCRIPT_NAME

[chemin+fichier] – le nom du script et le chemin de l'URL.

Exemple : /t3livre/index.php

QUERY_STRING

[requête] – la chaîne de caractères de la requête.

Exemple : id=26&tx_myext[myparam]=1234

TYPO3_SITE_SCRIPT

[fichier][?[requête]] – le fichier script appelé avec la chaîne de caractères de la requête.

Exemple : index.php?id=26&tx_myext[myparam]=1234

TYPO3_REQUEST_URL

[protocole]://[hôte][:[port]][chemin][?[requête]] – l'URL complète.

TYPO3_REQUEST_HOST

[protocole]://[hôte][:[port]]

Exemple : http://www.example.net:80

TYPO3_REQUEST_SCRIPT

[protocole]://[hôte][:[port]][chemin+fichier]

Exemple : http://www.example.net:80/t3livre/index.php

TYPO3_REQUEST_DIR

[protocole]://[hôte][:[port]][chemin]

Exemple pour un module backend : `http://www.example.net:80/t3livre/typo3/ext/cc_beinfo/mod1/`

TYPO3_SITE_URL

[protocole]://[hôte][:[port]][chemin] – chemin de l'URL menant au site Web TYPO3

Exemple : `http://www.example.net:80/t3livre/`

Fichiers et répertoires**TYPO3_DOCUMENT_ROOT**

[chemin]

Chemin absolu vers le répertoire racine du site Web.

Exemple : `/var/www`

SCRIPT_FILENAME

[chemin+fichier]

Exemple : `/var/www/t3livre/index.php`

Vous pouvez rendre les variables d'environnement visibles avec les extensions FE Debug/Info output (cc_feinfo)/Backend Environment Information (cc_beinfo).

GET et POST

Les données GET et POST sont disponibles à travers les variables `$GLOBALS['HTTP_POST_VARS']` et `$GLOBALS['HTTP_GET_VARS']`. Vous devriez néanmoins utiliser les fonctions suivantes :

`t3lib_div::_GET($var)`

`t3lib_div::_POST($var)`

`t3lib_div::_GP($var)`

Les fonctions `_GET()` et `_POST()` renvoient les données qui ont été transmises par GET ou POST. Dans la plupart des cas, la manière utilisée pour transmettre les données importe peu. C'est pourquoi vous pouvez utiliser la fonction `_GP()`. Elle recherche, d'après la clé passée en argument, d'abord dans les données POST et ensuite dans les données GET. Rappelez-vous que ces fonctions renvoient les données sans barre oblique : la fonction `$GLOBALS['TYPO3_DB']->quoteStr()` doit être appliquée aux données avant qu'elles ne soient écrites dans la base de données.

À l'intérieur des plugins, vous accédez aux paramètres via `$this->piVars[]`, pour que, normalement, vous n'ayiez pas à vous préoccuper des fonctions ci-dessus.

Fichiers

Lorsque vous manipulez des fichiers, notez que sur les systèmes Windows, les fichiers texte et les fichiers binaires sont traités différemment. Mais dans la plupart des cas, cela n'est pas voulu et peut mener à des dysfonctionnements. Pour éviter ceux-ci, vous devriez toujours ouvrir ces

fichiers en mode binaire pour accéder aux fichiers en écriture et en lecture. Le mode binaire est activé en ajoutant la lettre **b** au paramètre `mode` dans `fopen()`.

```
$fp = fopen ($filename, 'rb');
```

PHP fournit la fonction `tempnam()` pour créer des fichiers temporaires, mais cela peut causer des problèmes si le *Safe Mode* est activé pour PHP. Pour cette raison, TYPO3 propose la fonction `t3lib_div::tempnam()` qui enregistre les fichiers temporaires dans `typo3temp/`.

Le lancement de programmes externes est la source de problèmes : le chemin du programme doit être connu ou configuré ; les fichiers exécutables sous Windows ont le suffixe `.exe` et la fonction `is_executable()` ne fonctionne pas sur les systèmes Windows. Si vous voulez appeler Perl, par exemple, il y a plusieurs manières de procéder : `C:/perl/bin/perl.exe`, `/usr/bin/perl` ou `/usr/local/bin/perl`. La classe `t3lib_exec` essaie de résoudre le problème. L'appel suivant fournit le chemin complet vers l'interpréteur Perl :

```
$cmd = t3lib_exec::getCommand ('perl');
```

On peut alors accéder aux programmes sans configurer statiquement leur chemin.

7.4.13 Multilinguisme

Référence 104717

TYPO3 est disponible dans de nombreuses langues, à la fois dans le backend et dans le frontend. À tel point que les extensions que vous avez publiées dans le Repository pourraient être traduites, par exemple, en swahili par un traducteur inscrit sans que vous l'ayiez prévu. Depuis quand n'avez-vous pas publié quelque chose dans six langues différentes ?

Les textes des modules et les plugins dans d'autres langues sont contenus dans les fichiers correspondant au fichier d'exemple `locallang*.php`. Le Kickstarter vous aide en générant ce fichier lors de la création de l'extension. Comme vous pouvez le voir dans l'exemple qui suit, un fichier `locallang` contient un tableau `$LOCAL_LANG` qui contient à son tour des tableaux avec les traductions dans différentes langues.

```
$LOCAL_LANG = Array (
    'default' => Array (
        'todos_new' => 'Create new To-Do',
        'todos_update' => 'Update To-Do',
        'todos_target' => 'Target user',
        'todos_type' => 'Workflow',
    ),
    'dk' => Array (
        'todos_new' => 'Opret ny opgave',
        'todos_update' => 'Opdater opgave',
        'todos_target' => 'Mål-bruger',
        'todos_type' => 'Arbejdsgang',
    ),
    'de' => Array (
        'todos_new' => 'Neue To-Do Liste anlegen',
        'todos_update' => 'To-Do-Liste aktualisieren',
        'todos_target' => 'Zielbenutzer',
        'todos_type' => 'Workflow',
    ),
);
```

Au moment de l'exécution, les textes sont créés dans la langue correspondante via les clés ('todos_new', 'todos_update'). Ceci est implémenté dans les plugins par la méthode `$this->pi_getLL()` et dans les modules par la méthode `$LANG->getLL()`. Avec `$LANG->getLL('todos_update')`, par exemple, le texte « Update To-Do » est affiché si l'utilisateur backend actuel a choisi la langue anglaise ou si aucune traduction n'est disponible pour la langue sélectionnée.

Il est généralement admis que l'utilisation des textes des clés dans le code d'un programme est quelque peu fastidieuse, de même qu'entrer le texte courant dans les fichiers `locallang`. Dans ce cas, vous pouvez utiliser l'extension `ExtDevEval` (voir la section 7.12 *Outils pour le développeur*) pour extraire le texte du code et le remplacer avec les textes des clés.

Comme nous l'avons déjà mentionné, TYPO3 offre également un support pour les sites Web multilingues au sein d'une seule arborescence de pages, via la table `pages_language_overlay`. On attribue les éléments de contenu aux différentes langues définies dans la table `sys_language` en spécifiant la valeur du champ `sys_language_uid` de la table `tt_content`.

Il n'y a pas de support spécial pour ce concept pour les données des plugins. Mais l'expérience montre que les capacités multilingues sont rarement nécessaires pour les données des plugins. Le moyen le plus simple est généralement d'utiliser un SysFolder différent pour chaque langue pour les enregistrements du plugin. Bien entendu, il est aussi possible de créer les champs pour les langues requises dans la table. Cette procédure n'a d'intérêt que pour des applications particulières.

Une autre possibilité est d'utiliser le concept de `tt_content` dans votre propre table. Pour ce faire, créez un champ `sys_language_uid` correspondant en vous inspirant de la définition TCA de `tt_content`. Vous aurez ensuite un champ de sélection de la langue dans les enregistrements. Les enregistrements correspondant à la langue de la page courante sont alors choisis comme suit :

```
SELECT ... WHERE sys_language_uid='.$GLOBALS['TSFE']->sys_language_uid...
```

Si l'extension Static Info Tables est installée, vous accéderez au code ISO-639³ de la langue avec `$GLOBALS['TSFE']->sys_language_isocode`.

7.4.14 Codage des caractères

Avant de parler des caractéristiques spéciales de TYPO3 en rapport avec le codage des caractères, nous en posons les bases. Si vous êtes déjà familiarisé avec ces principes, vous pouvez tout de suite passer à la section suivante.

Les principes du codage de caractères

D'ordinaire, les gens ne se posent pas de questions en encodant des caractères, et ce qu'on entend par là n'est même pas clair en général. Après tout, cela reste très simple – vous appuyez sur la touche « ö » (o umlaut) de votre clavier, et cette lettre apparaît à l'écran. Que devez-vous savoir de plus ? Eh bien, il y a des années, vous auriez peut-être essayé de lancer votre machine à écrire à roue d'impression ou votre imprimante matricielle par la fenêtre, car elle

Référence 996712

³http://en.wikipedia.org/wiki/ISO_639

ne voulait tout simplement pas imprimer les umlauts, et imprimait à la place des caractères graphiques – nous voilà face à un problème de codage de caractères.

Il est généralement admis que lorsqu'une lettre telle que « ö » est sauvée, le caractère lui-même est sauvegardé. Mais cela n'est pas vrai. Le caractère « ö », par exemple, vient d'un jeu de caractères TrueType et est affiché à l'écran par le système d'exploitation. En fait, le « ö » est encodé et sauvé sous forme de nombre et le caractère « ö » est aussi représenté par un nombre dans le jeu de caractères par le système d'exploitation. L'aspect intéressant ici est que ces deux nombres peuvent être différents, mais que le « ö » continue à s'afficher à l'écran lorsque vous appuyez sur la touche « ö » (qui à son tour peut être représentée par un nombre différent).

Ainsi, il ne suffit pas de connaître la valeur (le nombre) d'un caractère. Vous ne saurez de quel caractère il s'agit qu'à partir du moment où vous connaissez le codage employé, c'est-à-dire le système utilisé pour associer un nombre aux caractères. Le premier codage standardisé était ASCII, qui utilise seulement sept bits (0-127) pour le codage, et qui ne contient même pas les umlauts allemands. Puisque la plus petite unité d'enregistrement de donnée est l'octet, avec 8 bits⁴, vous pourriez utiliser un bit supplémentaire et donc définir 128 caractères supplémentaires. Ceci a mené finalement au codage ISO-8859 qui définit les jeux de caractères suivants :

- ISO 8859-1 (Latin-1) – Ouest-européen
- ISO 8859-2 (Latin-2) – Est-européen
- ISO 8859-3 (Latin-3) – Sud-européen et espéranto
- ISO 8859-4 (Latin-4) – Baltique
- ISO 8859-5 – Cyrillique
- ISO 8859-6 – Arabe
- ISO 8859-7 – Grec
- ISO 8859-8 – Hébreu
- ISO 8859-9 (Latin-5) – Turc au lieu d'islandais, autrement le même que Latin-1
- ISO 8859-10 (Latin-6) – Nordique
- ISO 8859-11 – Thaïlandais
- ISO 8859-12 – Celtique (jamais choisi)
- ISO 8859-13 (Latin-7) – Baltique (remplace Latin-4 et -6)
- ISO 8859-14 (Latin-8) – Celtique
- ISO 8859-15 (Latin-9) – Ouest-européen avec le caractère Euro
- ISO 8859-16 (Latin-10) – Sud-européen avec le caractère Euro

Tous ces jeux de caractères ont en commun le US-ASCII comme codage des 128 premiers caractères. Les 128 caractères suivants diffèrent selon le codage utilisé. En Europe de l'Ouest, le codage ISO 8859-1 est normalement utilisé. Il s'appelle aussi Latin-A et contient les umlauts allemands par exemple. Mais l'alphabet grec n'est pas inclus dans ce jeu de caractères. Cet alphabet est défini dans ISO 8859-7 qui ne contient pas d'umlauts. Cela signifie qu'il n'est pas possible d'utiliser des umlauts et des lettres grecques dans le même texte.

La table suivante illustre ceci ; elle montre quatre caractères et compare les différents codages.

⁴Il est éventuellement possible d'utiliser la longueur de bits que vous voulez. En fait, d'autres longueurs de bits que le 8-bit sont utilisées dans différentes applications et processeurs spécialisés.

Caractère	HTML	Unicode en HTML	ISO 8859-1	ISO 8859-7	DOS CP 850	Unicode	Unicode UTF-8
ö	ö ;	ö ;	0xF6	-	0x93	U+00F6	0xC3B6
φ	&phi ;	φ ;	-	0xF6	-	U+0278	0xC9B8
÷	÷ ;	÷ ;	0xF7	-	0xF6	U+00F7	0xC3B7
„ ¹⁾	&bdquo ;	„ ;	-	-	-	U+201E	0xE2809E

Tableau 7.2:
Comparaison du
codage pour quelques
caractères

¹⁾Guillemet-virgule double inférieur

La table est peut-être un peu confuse au premier abord, mais sa signification devrait être plus claire après cet exemple⁵ :

Nach der Erfindung der Telegrafie benötigte man auch
hier eine Zeichenkodierung. Aus den ursprünglichen
Ideen des Engländers Alfred Brain entstand 1837 der
Morsecode.

Pour ceux qui ne connaissent pas l'allemand, voici la traduction :

Après l'invention du télégraphe, le codage de caractères était nécessaire. Le code
Morse a été créé en 1837, d'après les idées de l'Anglais Alfred Brain.

En plus des caractères définis dans US-ASCII, ce texte contient aussi des umlauts (ö, ü et ä).
Admettons que ce texte ait été écrit avec un éditeur de texte sur un système utilisant le codage
ISO 8859-1 et qu'il ait été sauvé dans un fichier. Si ce fichier est transféré sur un ordinateur
sur lequel la version allemande du système d'exploitation DOS est lancée, et qu'il y est ouvert
dans un éditeur, vous verrez alors ceci :

Nach der Erfindung der Telegrafie ben÷tigte man auch
hier eine Zeichenkodierung. Aus den urspr³nglichen
Ideen des Englönders Alfred Brain entstand 1837 der
Morsecode.

Que s'est-il passé ? Les données n'ont pas changé, le texte a juste été affiché en se basant sur
la codepage 850 de DOS. Le « ö » a été sauvé avec le codage ISO 8859-1, c'est-à-dire avec la
valeur 0xF6 (hexadécimal). Dans la codepage 850 de DOS, toutefois, la valeur 0xF6 est réservée
à un autre caractère, à savoir le « ÷ » comme vous pouvez le voir dans le tableau ci-dessus.

Si le texte est affiché par erreur avec le codage ISO 8859-7, vous obtiendrez le résultat suivant :

Nach der Erfindung der Telegrafie benφtigte man auch
hier eine Zeichenkodierung. Aus den ursprönglichen
Ideen des Englönders Alfred Brain entstand 1837 der
Morsecode.

⁵Source : http://en.wikipedia.org/wiki/character_encoding

Ici, la valeur 0xF6 produit la lettre grecque « phi », c'est pourquoi le texte ci-dessus ne contient toujours pas le caractère désiré. Vous devez donc toujours savoir quel codage a été utilisé pour les données. Pour cette raison, le codage est spécifié dans les en-têtes des pages HTML.

```
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

Comme vous le voyez sur le tableau, il est impossible d'utiliser les caractères « ö », « ÷ » et « φ » simultanément dans un texte. Pour résoudre ce problème, le système Unicode a été introduit afin d'inclure tous les caractères.

L'avantage est qu'il n'y a qu'un système de caractères qui contient tous les caractères dans le monde (en admettant qu'ils aient déjà été définis dans Unicode). Mais puisqu'il y a plus de 256 caractères différents (ce qui correspond à 1 octet), nous avons besoin de deux ou même de quatre octets pour identifier un caractère. Cela signifie que lorsque vous sauvez des données, vous consommez jusqu'à quatre fois plus d'espace. Puisque ceci est incompatible avec d'autres codages, qui sont généralement basés sur US-ASCII, UTF-8 a été introduit. UTF-8 est pour ainsi dire Unicode avec un codage à 1 octet (8 bits). Dans ce codage, US-ASCII est stocké avec un seul octet. Cela veut dire qu'UTF-8 utilise un nombre variable d'octets pour enregistrer les données, mais généralement un seul octet si les données consistent en du texte composé majoritairement de lettres de a à z ou de A à Z. Même si certaines applications ne supportent pas encore Unicode, il est prévisible que ce codage deviendra un standard et que, tôt ou tard, tous les autres disparaîtront. La pratique nous indique toutefois que c'est principalement la version UTF-8 du codage Unicode qui est utilisée pour enregistrer et transmettre des données, alors que le codage UCS-2 (2 octets) est fréquemment utilisé pour le traitement interne.

Différents codages de caractères dans TYPO3

Si vous ne le configurez pas différemment, TYPO3 utilise le codage ISO 8859-1 qui semble être adéquat pour la plupart des utilisateurs dans le monde occidental. Cependant, même les guillemets typographiques utilisés en Allemagne ("") ne peuvent être affichés avec ce codage. Microsoft Windows utilise le codage Windows-1252 qui est identique dans les grandes lignes à ISO 8859-1 mais qui contient les guillemets allemands. Ainsi, il n'y a aucun problème pour utiliser ces caractères lorsque vous écrivez du contenu.

On évite ce problème pour l'affichage dans le frontend en se rabattant sur le codage HTML des caractères spéciaux et en utilisant „. Mais lors de la sauvegarde, ce codage devrait aussi être utilisé, puisque les données de la base de données sont encodées avec ISO 8859-1. Dès que vous voudrez utiliser les données pour un affichage dans un autre format que HTML, des problèmes vont survenir.

À propos, TYPO3 utilise un codage différent d'ISO 8859-1 pour certaines langues dans le backend, de façon à pouvoir afficher les caractères pertinents. Si vous utilisez le grec comme langue dans le backend, le codage ISO 8859-7 sera utilisé pour entrer et sauver du contenu dans la base de données. Pour vous assurer que les éléments de contenu entrés sont reproduits correctement dans le frontend, vous devez configurer les sites en conséquence via TypoScript.

```
config.metaCharset = iso-8859-7
```

Pour standardiser et simplifier, il faudrait utiliser UTF-8 pour le codage des caractères dans TYPO3. Tous les navigateurs modernes supportent ce codage, de sorte qu'aucune conversion n'est nécessaire pour l'affichage. Si de vieux navigateurs comme Netscape 4 ou Internet Explorer 4 sont supportés dans le frontend, les pages doivent être délivrées séparément pour ces navigateurs. Sinon, vous devrez vous passer de UTF-8, puisque cela pourrait aussi vous causer des problèmes dans ce cas.

Vous pouvez utiliser UTF-8 à partir de la version 3.6 de TYPO3. Cependant, nous ne pouvons vous recommander l'utilisation d'UTF-8 avec cette version, car certains problèmes mineurs doivent encore être résolus. Cette section doit donc être considérée comme une perspective.

Pour activer UTF-8 pour TYPO3, vous devez configurer l'option `forceCharset` via l'outil d'installation, ou directement dans `typo3conf/localconf.php` :

```
$TYPO3_CONF_VARS['BE']['forced_charset'] = 'utf-8';
```

Le backend, et donc le stockage des données dans la base de données, supporte à présent le codage UTF-8. Il doit toujours être reconnu par la base de données. L'outil d'installation de TYPO3 ne peut pas encore faire cela automatiquement, et vous devez donc exécuter vous-même la configuration avec les commandes SQL correspondantes. Pour MySQL 4.0⁶, le codage d'une table est modifié comme suit :

```
ALTER TABLE table_name CHARACTER SET utf8;
```

L'option TypoScript suivante doit être activée dans le **setup** pour avoir un affichage encodé avec UTF-8 dans le frontend :

```
config.metaCharset = utf-8
```

De cette manière, UTF-8 est utilisé de manière universelle pour l'encodage des caractères en TYPO3, et les textes du fichier de langues `locallang` sont automatiquement transformés dans le codage *ad hoc* via les méthodes `getLL()` (BE) et `pi_getLL()` (FE) correspondantes. Évidemment, les gabarits HTML doivent eux aussi utiliser le codage correspondant ; surtout s'ils contiennent du texte, un encodage erroné dans le frontend mènerait dans le cas contraire à un affichage incorrect.

Il reste encore à savoir comment les données, c'est-à-dire le texte, seront traitées et à repérer ce qui doit être pris en compte. Des problèmes peuvent apparaître ici pour les raisons suivantes : puisque les textes encodés avec UTF-8 peuvent consister en plus d'octets que de lettres qu'ils symbolisent, des singularités sont attendues lors du traitement. Par exemple, la fonction PHP `strlen()` fournit la longueur du texte à partir du nombre d'octets requis pour le sauver. Avec les textes encodés avec UTF-8, cela peut provoquer une différence entre la taille réelle du texte et le résultat de la fonction `strlen()`. Pour cette raison, le codage utilisé doit être reconnu par la base de données. Par exemple, une recherche lancée dans laquelle aucune distinction n'est faite entre minuscules et majuscules peut être faussée si le codage UTF-8 est utilisé.

TYPO3 lui-même fournit la classe `t3lib_cs` pour convertir les différents encodages des caractères, puisque PHP ne contient pas les fonctions nécessaires dans chaque installation. Une

⁶Le support de base de données pour UTF-8 est disponible dans MySQL à partir de la version 4.

API pour les plugins utilisée de manière transparente, par exemple une fonction `strlen()` dans `tslib_pibase`, n'a cependant pas encore été introduite. Cette API est nécessaire pour commencer à développer des plugins, qui peuvent manipuler du texte indépendamment du codage. Il est probable que des solutions existent déjà à l'heure où vous tenez ce livre entre vos mains. De l'information mise à jour est disponible à la référence de cette section.

7.5 Programmer dans le frontend : les principes

Par frontend, nous entendons généralement l'extension CMS, qui se trouve dans le répertoire `typo3/sysexts/cms/`, et qui est accessible dans le répertoire principal d'un site Web TYPO3 via le lien symbolique `tslib`.

Des fonctionnalités peuvent être adjointes au backend ou au frontend par le développement d'extensions. Une large gamme d'extensions (plugins) est déjà offerte par le Kickstarter, ce qui vous permet de mettre en place les vôtres rapidement. Les extensions du frontend sont dans certains cas limitées à quelques lignes de TypoScript, mais elles peuvent aussi constituer une véritable application PHP impliquant plusieurs bases de données.

7.5.1 Frontend : restitution du contenu

Nous vous donnons ici un aperçu du processus de restitution de contenu d'une page, ce qui est en principe la tâche du script `tslib/index_ts.php`.

- Durant la phase d'initialisation, les constantes sont fixées, la connexion à la base de données est établie et les bibliothèques du frontend sont intégrées.
- L'objet global `$TSFE` de la classe `tslib_fe` est créé. Il contrôle le processus de restitution de contenu.
- L'objet pour l'authentification d'un utilisateur frontend et pour la gestion de session est créé.
- Si un utilisateur backend est actif, des fonctions complémentaires telles que l'édition dans le frontend et le panneau d'administration sont initialisées.
- L'ID et le type de la page appelée sont déterminés. Dans le même temps, les permissions d'accès sont vérifiées.
- Le moteur TypoScript de gabarit (TSFE, TypoScript Template Engine) est initialisé.
- Le cas échéant, la page est lue à partir du cache dans la base de données.
- Le tableau `config` est initialisé à partir du Setup TypoScript (`config.*`).
- Les données principales du TCA sont lues.
- La langue est déterminée (cf. section 7.4.13).
- Les données transmises, telles que celles d'un formulaire email, sont traitées.
- Si la page n'est pas lue à partir du cache, elle est restituée par la configuration TypoScript et enregistrée dans le cache.
- Les objets (cObject) qui ne doivent pas être mis en cache sont restitués et affichés dans le frontend : `PHP_SCRIPT_INT`, `USER_INT`, `PHP_SCRIPT_EXT`.

- La page restituée (\$TSFE->content) est affichée avec `echo()`.
- Les données de session des utilisateurs frontend sont sauvegardées.
- Les logs sont enregistrés.
- Le cas échéant, la redirection vers une adresse URL (`jumpurl`) est effectuée.
- Au cas où la page est demandée par un utilisateur backend, un cadre contenant le message « preview » est inséré.
- Un fichier HTML statique est enregistré si l'option correspondante a été activée dans le panneau d'administration.
- Le panneau d'administration est inséré si la configuration l'a spécifié.
- Si une extension de débogage est installée, elle est appelée pour modifier l'affichage en sortie.

7.5.2 API frontend

De même que pour `t3lib`, des bibliothèques et des objets supplémentaires sont disponibles dans le frontend. La restitution du contenu se base sur la structure d'objet suivante :

```
$TSFE (tslib_fe)
|
|-----> fe_user (tslib_feUserAuth)
|
|-----> sys_page (t3lib_pageSelect)
|
|-----> cObj (tslib_cObj)
|
|-----> myPluginObj (extends tslib_pibase)
|
|-----> cObj (tslib_cObj)
```

Dans la majorité des cas, une extension frontend est un plugin qui étend la classe de base `tslib_pibase`. À partir d'un plugin, vous accédez donc directement aux classes et aux objets suivants :

`tslib_fe`

Le TypeScript frontend (TSFE) est disponible en tant qu'objet global pour les plugins via `$GLOBALS['TSFE']`.

`tslib_cObj`

Est disponible en tant qu'objet pour les plugins via `$this->cObj`. Il contient des méthodes pour restituer les objets TypeScript tels que `TEXT` et `IMAGE`. De plus, les fonctions `stdWrap` et `parseFunc` sont des méthodes de cette classe.

`tslib_pibase`

Les plugins sont une extension de cette classe. Cette dernière fournit un ensemble de fonctions utiles aux plugins.

`t3lib_pageSelect`

Fonctions sur les pages. Elles peuvent être appelées dans le FE via l'objet `$GLOBALS['TSFE']->sys_page`.

t3lib_div

La série de fonctions de `t3lib_div` est aussi disponible dans le frontend.

Dans la suite, nous présentons une à une ces bibliothèques.

7.5.3 TypeScript frontend (TSFE)

Le TypeScript frontend (TSFE) est un objet disponible pour les plugins via la variable globale `$TSFE`. Il contient de l'information, des méthodes et des objets. Comme mentionné précédemment, le TSFE est l'objet central dans le processus de restitution du contenu. Pour la plupart des plugins, les objets `tslib_pibase` et `cObj` couvriront vos besoins.

Vous trouverez ci-après une partie des données et des objets du `$TSFE`.

`$TSFE->id`

uid de la page courante

`$TSFE->page[]`

Tableau contenant l'enregistrement de la page courante

`$TSFE->sys_page`

Objet avec différentes méthodes s'appliquant aux pages

`$TSFE->additionalHeaderData[]`

Tableau contenant des données supplémentaires à destination de l'en-tête HTML

`$TSFE->sys_language_uid`

ID de la langue courante

`$TSFE->tmpl`

Objet de gabarit TypeScript

`$TSFE->tmpl->setup[]`

Objet contenant l'ensemble de la configuration TypeScript

`$TSFE->pSetup[]`

Objet contenant la configuration TypeScript de l'objet de la page

`$TSFE->config[]`

Tableau de configuration (TS config)

`$TSFE->register[]`

Registre TypeScript

`$TSFE->cObj`

Objet central ; un objet `cObject` est disponible dans les plugins via `$this->cObj`.

`$TSFE->fe_user`

Utilisateur frontend courant (object)

L'objet `$TSFE` fournit un certain nombre de méthodes spécialement réservées aux plugins :

`getStorageSiterootPids()`

Renvoie un tableau contenant les valeurs `_SITEROOT` et `_STORAGE_PID` qui spécifient respectivement les ID de la page et de la page racine du site Web, ainsi que l'endroit où les enregistrements doivent être stockés.

getPagesTSconfig()

Cette méthode renvoie le tableau TSConfig de la page en fonction du rootline

setJS()

Spécifie le code JavaScript, qui sera inséré dans l'en-tête HTML

setCSS()

Spécifie les données CSS qui seront insérées dans l'en-tête HTML

uniqueHash()

Génère une valeur de hachage unique

set_no_cache()

Active l'option pour que la page ne soit pas cachée

set_cache_timeout_default()

Fixe la durée après laquelle la page est régénérée dans le cache

L'extension **FE Debug/Info output (cc_feinfo)** vous fournit un plugin pour afficher dans le frontend les valeurs courantes de l'objet TSFE et d'autres données utiles. Ceci vous aide à déboguer ou à visualiser l'ensemble des données disponibles.

7.5.4 cObject, tslib_cObj

Un cObject (Content Object) est un objet TypeScript, comme par exemple TEXT, IMAGE, HMENU. Ces objets sont instanciés en PHP par la classe **tslib_cObj** (`class.tslib_content.php`). De plus, vous y retrouvez les fonctions telles que **stdWrap** ou **parseFunc**.

Dans les plugins, vous accédez à l'instance de **tslib_cObj** via `$this->cObj`. Cette référence à l'objet est définie automatiquement lors de l'initialisation du plugin. Voici un aperçu de l'API de **tslib_cObj** :

data[]

Permet d'accéder à l'enregistrement courant (normalement de la table **tt_content**).

cObjGetSingle()

Restitue le cObject en fonction du nom passé en argument (TEXT, IMAGE, ...) et de la configuration TypeScript. Vous trouverez plus de détails à ce sujet à la section suivante.

stdWrap()

La fonction standard pour les enveloppes (*wrap*). En appliquant les paramètres de votre TypeScript, cette fonction vous offre une large gamme de possibilités. Vous avez déjà pu en juger lorsque nous avons abordé l'exemple « un compteur de visiteurs en 20 minutes » (cf. section 7.1).

enableFields()

Crée une clause SQL WHERE qui sélectionne uniquement les enregistrements valides dans le FE. On prend ici en compte les permissions d'accès et les champs **Lancement** et **Arrêt**.

DBgetUpdate()

Crée une commande SQL Update pour une table en tenant compte de la configuration de la table dans le tableau TCA.

DBgetInsert()

Comme pour `DBgetInsert()`, mais pour l'insertion d'enregistrements.

Arguments

Dans leur mise en pratique en PHP, les objets de contenu traitent de la même manière le passage d'arguments :

```
function cImage($file,$conf)
function stdWrap($content,$conf)
function typoLink($linktxt, $conf)
```

Le premier argument contient une valeur (par exemple une chaîne de caractères), qui doit être traitée. Le second argument est la variable `$conf`, qui détermine le comportement de la méthode.

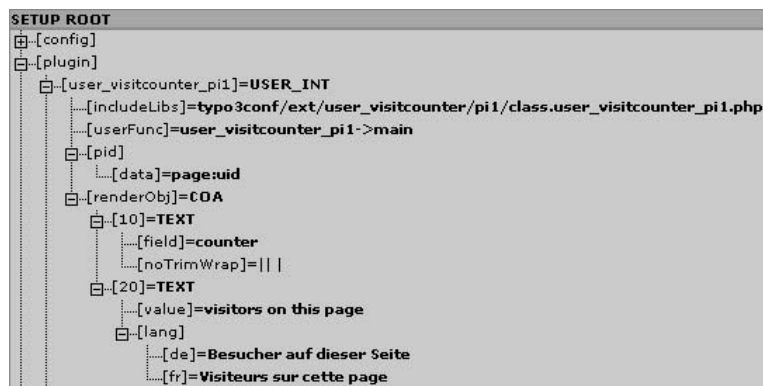
Les plugins respectent aussi cette convention, puisqu'ils sont instanciés via `USER` ou `USER_INT` et qu'une méthode prend comme arguments les variables `$content` et `$conf`. Cependant, dans la majorité des cas, la variable `$content` n'est pas utilisée et peut être ignorée.

Tableau \$conf

Le tableau `$conf` représente l'interface entre TypoScript et PHP. D'une part, il permet de configurer le code PHP et d'autre part, en utilisant les `cObjects` à l'intérieur du code PHP, vous pouvez modifier ou changer leur comportement. Nous avons déjà illustré ce dernier point dans la restitution graphique du compteur de visiteurs (cf. section 7.1).

En utilisant l'exemple du compteur de visiteurs, nous allons à présent montrer le fonctionnement du tableau `$conf`. À la figure suivante, le TypoScript Object Browser vous présente la configuration TypoScript du compteur de visiteurs.

Figure 7.28:
Configuration TS dans
le TypoScript Object
Browser



Comme vous le constatez, le TypoScript a été converti en une liste hiérarchique qui a déjà été décrite en détail précédemment.

La configuration TS est passée en argument à l'objet `USER_INT` dans `plugin.user_visitcounter_pi1`. Voici `$conf` tel qu'il apparaît à l'aide de la fonction `debug()` :

includeLibs	typo3conf/ext/user_visitcounter/pi1/class.user_visitcounter_pi1.php
userFunc	user_visitcounter_pi1->main
pid.	data page:uid
renderObj	COA
renderObj.	10 TEXT
	10. field counter
	noTrimWrap
	20 TEXT
	value visitors on this page
	lang. de Besucher auf dieser Seite
	fr Visiteurs sur cette page

Figure 7.29:
Affichage de `$conf`
par `debug()`

L'objet `USER_INT` n'accepte que les deux arguments `includeLibs` et `userFunc`, qui définissent la méthode à appeler. Dans notre exemple, `USER_INT` appelle la méthode `user_visitcounter_pi1->main` et passe en argument le tableau `$conf` qu'il a lui-même reçu.

Comme vous le remarquez à la figure 7.29, l'objet `USER_INT` transfère les clés `renderObj` et `renderObj.` (avec le point à la fin). Les clés sans le point sont initialisées via le signe `<=>` dans TS, généralement dans le but de définir le type d'objet. La même clé avec le point `<.>` contient la configuration de l'objet.

```
renderObj = COA
renderObj {
// dans l'objet renderObj. (avec le point)
```

La seconde possibilité est que le signe `<=>` fixe une valeur au lieu du type d'objet. Ce concept est illustré dans l'exemple ci-dessus par le paramètre `pid`. Dans le tableau de configuration TS, seul `pid.` (avec un point) est disponible. Clairement, `pid` n'est pas défini avec le signe `<=>`. Observons la ligne TypoScript correspondante :

```
pid.data = page:uid
```

Dans le cas du paramètre `pid`, vous ne devez pas le définir car il est de type `stdWrap`. Il est donc pris en charge par le code PHP du plugin.

```
$pid = intval($this->cObj->stdWrap($conf['pid'],$conf['pid.']));
```

La fonction `stdWrap()` détermine la valeur du `pid`. Si vous observez les paramètres, vous reconnaîtrez les concepts de `$content` et `$conf`. Dans la configuration ci-dessus, `$content` et `$conf['pid']` sont vides. La configuration est transférée avec `$conf['pid.']` et contient alors `data = page:uid`. Cela signifie que la fonction `stdWrap()` reçoit une valeur vide, mais qu'elle génère la valeur à renvoyer grâce à la configuration de `data`.

Les choses auraient été différentes avec le TypoScript suivant :

```
pid = 21
```

Dans ce cas, `stdWrap()` aurait reçu la valeur 21 (`$content`), mais pas de configuration (`$conf`). `stdWrap()` renvoie alors simplement 21.

Dans l'exemple suivant, la valeur est transférée, mais elle est remplacée par `override`.

```
pid = 21
pid.override.data = register: user_visitcounter_pid
```

Comme vous avez pu le constater, la combinaison du TypoScript et de PHP offre plusieurs façons de rendre le code PHP flexible, via la configuration de paramètres.

7.5.5 Restitution des cObjects par PHP

Comme l'exemple du compteur de visiteurs l'a montré, l'affichage d'un plugin peut aussi être produit par des cObjects. Même si un plugin génère une grande partie de l'affichage par lui-même, avec la mise en forme contrôlée par CSS, il peut être utile de contrôler certaines parties par TypoScript : par exemple, pour l'affichage des images.

```
$outputHTML = $this->cObj->cImage($file, $conf['image.']);
```

La méthode `cImage()` (IMAGE) restitue une image dans le frontend pour laquelle le nom du fichier est passé dans `$file`, en utilisant `$conf['image.']` de la configuration TS. De cette manière, vous pouvez par exemple modifier la taille de l'image avec TS. De plus, la balise `<img>` est générée.

Dans le compteur de visiteurs, tout l'affichage est contrôlé par un cObject. Cette opération est réalisée par le transfert de la configuration TS `renderObj` à la méthode `cObjGetSingle()`. Cette méthode restitue le contenu en fonction de la configuration, ou appelle la fonction de restitution de contenu pour le cObject en question. Par conséquent, vous disposez d'une grande liberté pour la restitution du contenu. Il est même possible de définir un `renderObj` en tant qu'objet USER pour générer la sortie via un script externe.

La méthode détermine le type d'objet à partir de `$conf['renderObj']`, ce qui signifie que `renderObj` peut être de n'importe quel type (TEXT, IMAGE, COA, ...).

```
// restitution du compteur par le TypoScript renderObj
$lCObj = tslib_div::makeInstance('tslib_cObj');
$lCObj->setParent($this->cObj->data, $this->cObj->currentRecord);
$lCObj->start($row, $table);
$content = $lCObj->cObjGetSingle($conf['renderObj'],
                                $conf['renderObj.']);
```

Comme vous le remarquez, `$this->cObj`, qui est automatiquement disponible au sein des plugins, n'est pas utilisé. En effet, l'objet travaille avec les données qui appartiennent toujours à un cObject. En général, ces données sont les champs de l'enregistrement courant, qui est d'habitude contenu dans la table `tt_content`.

L'exemple suivant montre une partie du TypoScript de la configuration de l'élément de contenu en-tête. Grâce à `field = subheader` et à la configuration TS, vous pouvez directement accéder au champ `subheader` de l'enregistrement `tt_content` courant.

```
tt_content.header = COA
tt_content.header {
    ...
    20.1 = TEXT
    20.1.field = subheader
}
```

Dans le compteur de visiteurs, `$this->cObj->cObjGetSingle()` n'aboutirait à rien avec la configuration TS suivante, puisque les données ne proviennent pas de la table `user_visitcounter`, mais de l'enregistrement `tt_content`, qui insère le plugin dans la page.

```
10 = TEXT
10.field = counter
```

Pour cette raison, on crée une instance locale de `tslib_cObj`. Ensuite, on ajuste avec `$ICObj->start($row, $table)` l'enregistrement courant en reprenant le dernier enregistrement du compteur de visiteurs qui a été lu. Le TS ci-dessus donnera alors le résultat voulu.

7.5.6 tslib_pibase

TYPO3 met à disposition la classe de base `tslib_pibase` pour les plugins du frontend. Les plugins créés par le Kickstarter sont basés sur `tslib_pibase`. Un plugin ne doit pas nécessairement utiliser cette classe, mais cette dernière contient des méthodes spécialement destinées aux plugins. Parmi les méthodes présentes, on notera celles relatives à la création correcte des liens et au traitement des paramètres.

Voici la gamme des fonctions fournies par `tslib_pibase` :

Reconnaissance des paramètres

`tslib_pibase` reconnaît les paramètres qui sont passés au plugin. La section suivante donne plus de détails à ce sujet.

Méthodes pour les liens

Méthodes pour générer les liens avec ou sans paramètres. Elles sont aussi décrites plus en détail à la section suivante.

Méthodes pour les listes d'enregistrements

Le Kickstarter peut créer des plugins qui listent déjà des enregistrements et les affichent dans la vue détaillée du backend. `tslib_pibase` fournit les méthodes de base à cet effet.

Stylesheet et CSS

Un certain nombre de méthodes sont disponibles pour l'utilisation des CSS. Celles-ci tiennent compte de l'espace de nommage du plugin.

Édition frontend

Méthode pour générer les éléments d'édition du FE (icônes) pour les enregistrements.

Localisation, langues

Support des plugins multilingues sur base du fichier de langues `locallang`.

Accès à la base de données

Méthodes pour accéder à la base de données.

Cache

Support du cache pour les plugins avec des paramètres.

Flexforms

Support pour le traitement des données des Flexforms.

Codage de caractère

Méthodes pour traiter les chaînes de caractères, en prenant en compte le codage de caractères.

7.5.7 Liens et paramètres dans les plugins

En générant les liens au sein des plugins, deux principes doivent être respectés: d'une part, les liens doivent avoir un format correct, en l'occurrence, les liens vers une page de TYPO3 doivent comprendre l'ID et le paramètre **type**. D'autre part, les paramètres GET utilisés doivent respecter la convention de nommage de l'extension pour éviter les conflits avec d'autres plugins.

En outre, il existe plusieurs manières d'afficher les URL dans TYPO3. La première qui vaut la peine d'être mentionnée est celle qui se présente sous une forme non encodée :

```
index.php?id=123&tx_example_pi1[showUid]=456
```

En plus, l'option `simulateStaticDocuments` ou l'extension **SpeakingURLs** vous offrent d'autres possibilités de codage qui augmentent la lisibilité et améliorent le référencement de votre site par les moteurs de recherche.

La classe de base `tslib_pibase` de plugin vous donne un cadre pour la création de vos URL et pour la gestion des paramètres de vos plugins. Vous ne devez dès lors plus vous soucier du format.

L'exemple de code suivant montre les éléments typiques d'un plugin généré par le Kickstarter en ce qui concerne la gestion des paramètres.

```
require_once(PATH_tslib.'class.tslib_pibase.php');
class tx_example_pi1 extends tslib_pibase {
    var $prefixId = 'tx_example_pi1';
    ...
    function main($content,$conf) {
        $this->conf = $conf;
        $this->pi_setPiVarDefaults();
    }
    ...
}
```

Tout d'abord, les paramètres sont gérés par `$this->piVars[]`. Durant la phase d'initialisation, ce tableau est rempli par les paramètres GET et POST correspondant à `$this->prefixId`. Le constructeur de la classe de base `tslib_pibase` assure cette tâche automatiquement.

```
function tslib_pibase() {
    if ($this->prefixId) {
        $this->piVars = tslib_div::GPararrayMerged($this->prefixId);
    }
}
```

Si le plugin est appelé via la méthode `main()`, les valeurs par défaut pour les paramètres sont fixées par TypoScript (via `_DEFAULT_PI_VARS`) à l'aide de la méthode `pi_setPiVarDefaults()`. Mais ces valeurs sont remplacées par celles qui existent déjà dans `$this->piVars[]`.

Les possibilités suivantes existent pour la création des URL ou des liens :

- URL avec seulement l'ID et le type de la page
- URL avec des paramètres spécifiques
- URL avec des paramètres sélectionnés à partir de `$this->piVars[]`
- URL avec tous les paramètres `$this->piVars[]`
- URL avec tous les paramètres `$this->piVars[]` et les paramètres supplémentaires de remplacement

Dans tous ces cas, des méthodes de la classe de base `tslib_pibase` sont à votre disposition. Ces dernières reposent elles-mêmes sur les méthodes de liens de `tslib_cObj` qui sont mentionnées ici par souci d'exhaustivité.

```
$this->cObj->getTypoLink($label, $params, $urlParameters=array(), $target='')
$this->cObj->getTypoLink_URL($params, $urlParameters=array(), $target='')
$this->cObj->typoLink($linktxt, $conf)
```

Normalement, les méthodes suivantes de la classe de base `tslib_pibase` devraient vous suffire.

`pi_getPageLink()`

Cette méthode crée des URL pour un ID spécifique de page. Vous pouvez spécifier un tableau avec les paramètres de l'URL.

`pi_linkToPage()`

Comme pour `pi_getPageLink()`, mais un lien est créé (balise `<a>`) autour de la chaîne de caractères passée en argument.

Par opposition aux deux méthodes précédentes, les méthodes suivantes prennent aussi en compte le cache, comme cela sera expliqué à la section suivante.

`pi_linkTP()`

Comme pour `pi_linkToPage()`, à cette différence près que le cache est pris en considération.

`pi_linkTP_keepPIvars_url()`

Crée une URL, dans laquelle tous les paramètres de `piVars` sont inclus. Les paramètres peuvent être remplacés ou ajoutés.

`pi_linkTP_keepPIvars()`

Idem que pour la méthode précédente, mais crée un lien.

`pi_list_linkSingle()`

Crée un lien avec le paramètre `showUid`, pour afficher un record en particulier.

Une URL simple pour la page courante, à utiliser dans un formulaire par exemple, peut être générée comme suit:

```
$url = $this->pi_getPageLink($GLOBALS['TSFE']->id);
```

Le code suivant crée un lien pour naviguer à travers une liste. `pi_linkTP_keepPIvars()` est utilisé puisque les paramètres courants doivent être maintenus. Le paramètre `pointer` est passé et remplace donc la valeur existante, si elle existe, dans `$this->piVars[]`.

```
$browseNext = $this->pi_linkTP_keepPIvars(
    $this->pi_getLL('pi_list_browseresults_next', 'Next >', TRUE),
    array('pointer'=>$pointer+1));
```

L'utilisation des méthodes de liens via `piVars` et `tslib_pibase` demande une certaine discipline. Mais une fois que vous vous y êtes habitué, vous apprendrez à apprécier la limpidité de ce concept. De plus, il est impératif d'utiliser ce cadre de développement si vous voulez que votre plugin soit conforme à l'API. Dans le cas contraire, il y aura des conflits avec d'autres applications. Son utilité principale est sa compatibilité avec l'option `simulateStaticDocuments`, l'extension `SpeakingURLs` et d'autres solutions similaires. Enfin, comme il est décrit plus bas, vous avez la possibilité de mettre en cache des plugins.

7.5.8 USER, USER_INT, cache et paramètres

Le frontend utilise une mémoire tampon (cache), pour les pages déjà restituées. Une fois sauvegardées dans le cache, les pages ne doivent plus être reconstituées, elles peuvent être servies directement. TYPO3 dispose d'une large gamme d'options pour paramétrer le cache.

Tout d'abord, le comportement du cache est défini grâce aux paramètres suivants dans la configuration TS :

```
config.no_cache = 1
    Désactive le cache
config.cache_period = 3600
    Détermine la période (en secondes) après laquelle une entrée du cache expire
config.cache_clearAtMidnight = 1
    Vide le cache à minuit
```

De plus, vous pouvez spécifier individuellement, dans l'enregistrement de la page, si cette dernière doit être cachée ou non (**Sans cache**). Vous pouvez aussi définir la durée de validité du cache, qui détermine la fréquence à laquelle la page doit être complètement reconstruite (menu **Expiration du cache**, correspondant à `cache_period`).

Le comportement du cache peut aussi être contrôlé par du code PHP à partir des plugins. Si votre plugin est intégré en tant qu'objet `USER`, qui est normalement caché, vous pouvez forcer la page à être reconstruite toutes les 15 minutes en spécifiant :

```
$GLOBALS['TSFE']->set_cache_timeout_default(60*15);
```

Vous pouvez aussi empêcher la page d'être cachée durant le processus de restitution en activant le script suivant :

```
$GLOBALS['TSFE']->set_no_cache();
```

Mais cette méthode pour éviter d'enregistrer la page dans le cache est rarement nécessaire, ou n'a tout simplement pas de sens. Ce qui est beaucoup plus fréquent, c'est que le plugin doive réagir à la soumission d'un formulaire. Un objet `USER` n'en est pas capable, puisqu'il n'est même pas consulté si la page est cachée. Une solution à ce problème est d'activer le paramètre `no_cache=1`, pour que le frontend ne fasse plus appel au cache pour cette page, mais la reconstruise complètement. Dans les formulaires, vous pouvez simplement inclure ce paramètre en tant que champ caché.

```
<input type="hidden" name="no_cache" value="1">
```

Le cache n'est pas toujours possible ou n'a pas de sens, par exemple pour un plugin qui doit prendre en compte des données passées en arguments ou à des paramètres. Pour cette raison, les plugins peuvent être intégrés dans le cache de la page en tant qu'objets sans cache, à l'aide de `USER_INT`. L'avantage dans ce cas est la rapidité, puisque seules les parties dynamiques de la page doivent être restituées.

TYPO3 offre encore d'autres possibilités pour cacher le résultat des plugins via des paramètres. Mais cette fonctionnalité doit être prise en compte durant la programmation et doit être bien pensée, puisqu'une page séparée est cachée pour chaque combinaison des paramètres.

Supposons que nous avons un plugin qui affiche la liste des données d'une table mise à jour à quelques jours d'intervalle seulement. C'est donc un bon candidat pour le cache. Les enregistrements affichés sont sélectionnés par trois variables, chacun comportant 2 états. Par conséquent, il y a (2^3) listes. De plus, le tri peut être activé sur quatre colonnes, ce qui donne un total de 128 (2^7) vues différentes.

La liste a une longueur moyenne de 100 lignes, divisées en pages de 20 lignes. Cela signifie qu'il y a 640 ($128 (128 \times 5)$) variations de pages possibles dans le cache.

Si nous ajoutons deux paramètres supplémentaires, le premier avec cinq options et le second avec huit, nous arrivons à 25 600 variations possibles, qui ne seront vraisemblablement pas toutes appelées ou créées.

Comme vous le voyez, un plugin génère facilement des millions de pages cachées, ce qui charge lourdement le serveur de base de données. Il est difficile de faire une recommandation ici. Une application consommatrice de ressources, avec seulement quelques options, peut tirer parti du cache, tandis qu'un plugin qui affiche des données courantes n'est pas caché, bien sûr.

Voici un exemple de plugin pour démontrer les possibilités du cache. Le plugin génère l'affichage suivant :

Options:

☒ opt1
☒ opt2
☐ opt3

Rendering time: 18:07:35

[Link with Parameter](#)
[Link with Parameter opt3=1](#)

☒ GET/POST Vars
 ☐ Global Objects
 ☒ TSFE (excerpt)
 ☐ PHPinfo
☐ User
 ☐ Local Variables
 ☐ Environment

☐ Constants
 ☐ Global Variables
 ☐ Debug info

GET Variables:

Array (3) GET Variables

String (2) id	28
Array (2) tx_linkexample_pi1	String (1) opt1 1 String (1) opt2 1
String (10) cl lash	4901bb2eda

Figure 7.30:
Affichage par le
plugin linkexample

Le plugin comprend un formulaire avec trois options ; il donne l'heure lorsque la restitution est terminée, et deux liens. Pour finir, le plugin `cc_feinfo` est intégré pour que les paramètres passés soient affichés.

La page vient d'être appelée via le lien « Link with Parameter ». En dessous de « GET Variables », vous voyez les paramètres qui ont été transférés via le lien :

```
http://www.example.org/index.php?id=28&tx_linkexample_pi1[opt1]=1&tx_linkexample_pi1[opt2]=1&cHash=4901bb2eda
```

Vous retrouvez d'abord l'ID de la page, ensuite les deux paramètres `opt1` et `opt2` pour le plugin, qui ont été automatiquement ajoutés via `tslib_pibase`, et qui sont à présent disponibles dans `$this->piVars[]`. C'est dans le dernier paramètre `cHash` que réside le secret du cache des plugins avec paramètres.

Une entrée dans le cache est basée principalement sur l'ID de la page et le paramètre `type`. Si `cHash` est activé, les paramètres transférés sont inclus et on vérifie si la valeur de `cHash` correspond aux paramètres courants.

De cette manière, on évite que la page fasse délibérément l'objet d'une série d'appels avec des valeurs de hachage falsifiées.⁷

Retournons à présent à l'exemple de plugin. Le plugin est enregistré dans le fichier `ext_local-conf.php` de l'extension.

```
t3lib_extMgm::addPItoST43($_EXTKEY,
    'pi1/class.tx_linkexample_pi1.php',
    '_pi1',
    'list_type',
    1 /*cached*/);
```

Le dernier paramètre de `t3lib_extMgm::addPItoST43()` détermine si le plugin doit être intégré en tant qu'objet `USER_INT` (0) ou `USER` (1). Puisque les objets `USER_INT` ne sont normalement pas cachés, la valeur du paramètre doit être fixée ici à 1. Cette configuration n'est pas suffisante pour que la page soit cachée. Il faut encore spécifier le paramètre `cHash` qui est généré par la fonction de lien de la classe `tslib_pibase` (en définitive, par la méthode `cObj->typoLink()`). Si, par exemple, le paramètre `$cache` est fixé à `TRUE` lors de l'appel à la méthode `pi_linkTP_keepPIvars()`, le paramètre `cHash` est ajouté à l'URL.

```
function pi_linkTP_keepPIvars($str,
    $verrulePIvars=array(),
    $cache=0,
    $clearAnyway=0,
    $saltPageId=0)
```

Vous devez donc décider au sein du plugin si la page doit être cachée ou non, en fonction des paramètres, et ajuster la variable `$cache` en conséquence. Si vous voulez cacher tout ce qui est généré par le plugin, vous devez toujours sélectionner `TRUE` pour la valeur de `$cache`.

⁷La fonction de création de valeurs de hachage se sert du paramètre `$TYPO3_CONF_VARS['SYS']['encryptionKey']`. Ce dernier ne devrait pas être vide pour des raisons de sécurité (voir la section sur l'outil d'installation).

Mais n'oubliez pas que cela peut surcharger la base de données, selon le nombre de paramètres concernés.

Le tableau `pi_autoCacheFields[]` de la classe `tslib_pibase` permet d'activer par défaut le cache pour certains paramètres. Cette fonctionnalité est utilisée ci-après dans l'exemple de plugin. Nous nous limitons à montrer la méthode `main()` en laissant de côté l'en-tête et le pied de page du fichier du plugin.

```
function main($content,$conf) {

$this->conf=$conf;
$this->pi_setPiVarDefaults();

    // If set caching is disabled
$this->pi_USER_INT_obj=0;
```

La valeur de la variable `pi_USER_INT_obj` doit être fixée à `FALSE` pour activer le cache dans `tslib_pibase` et générer la valeur de `cHash`.

```
    // enable auto caching
$this->pi_autoCacheEn = 1;
    // register parameter for auto caching
$this->pi_autoCacheFields = array(
    'opt1' => array('list' => array(0,1)),
    'opt2' => array('list' => array(0,1)),
);
```

La fonction auto-cache est activée. Le tableau `pi_autoCacheFields[]` reprend les paramètres pour lesquels le cache est permis. Si tous les paramètres utilisés pour créer une URL sont disponibles dans ce tableau (`opt1,opt2`), et si leur valeur est définie, la valeur de `cHash` est générée. Sinon, le paramètre `no_cache=1` est rajouté pour que les pages soient complètement reconstruites, et pour que le plugin puisse prendre en compte les paramètres.

Les valeurs possibles sont définies par les tableaux `list` ou `range`. Alors que `list` contient une série de valeurs, `range` est défini par une borne inférieure et une borne supérieure.

```
$params = array('opt1','opt2','opt3');

    // build options form
$options = '';
foreach ($params as $opt) {
    $options.= '<input type="checkbox" '.
        'name="'. $this->prefixId.'['.$opt.']" '.
        'value="1" '.
        ($this->piVars[$opt]?'checked="checked"':'').
        '> '.$opt.'<br />';
}

$content = '
<h3>Options:</h3>
<form action="'.
    htmlspecialchars($this->pi_getPageLink($GLOBALS['TSFE']->id)).
    '" method="post">
```

```

        '.$options.'
        <input type="hidden" name="no_cache" value="1">
        <input type="submit" name="'.$this->prefixId.'_submit_button'
        value="Submit">
    </form>
    <br />
';

```

L'exemple de plugin contient les trois paramètres **opt1**, **opt2** et **opt3**. Le code ci-dessus crée un formulaire qui assigne des valeurs à ces paramètres. Le formulaire est envoyé à la page courante. Le paramètre **no_cache** est inséré en tant que champ caché du formulaire.

```

// show the current time - will not change with cached pages
$content.= 'Rendering time: '.date('H:m:s',time());
$content.= '<br /><br />';

```

L'heure est affichée pour visualiser immédiatement l'effet du cache. Si l'heure change à chaque chargement de la page, cela signifie que la page a été reconstruite et qu'elle ne provient pas du cache. Gardez à l'esprit que le cache peut être désactivé, pour différentes raisons, si vous vous êtes identifié dans le backend.

```

// link with current piVars set by options form
$content.= $this->pi_linkTP_keepPIvars('Link with Parameter');
$content.= '<br />';

```

Un lien est créé. Il envoie les valeurs courantes de **piVars** comme arguments à la page courante. Il serait possible de spécifier ici la variable **\$cache**, mais cela n'est pas nécessaire, car la fonction auto-cache a été activée. Si les paramètres **opt1** et/ou **opt2** ont une valeur, un lien est créé avec le paramètre **cHash**. Si le paramètre **opt3** est aussi présent, c'est **cache=1** qui est ajouté à l'URL, parce que **opt3** ne se trouve pas dans le tableau **pi_autoCacheFields[]**.

```

// overrule opt3=1 which will force no_cache=1
// (opt3 is not set in pi_autoCacheFields)
$overrule = array('opt3' => 1);
$content.= $this->pi_linkTP_keepPIvars('Link with Parameter opt3=1', $overrule);
$content.= '<br />';

```

Dans l'appel à la méthode **pi_linkTP_keepPIvars()**, le tableau **\$overrule**, contenant le paramètre **opt3** est passé en argument. Ce paramètre est donc ajouté au lien, qu'il soit ou non contenu dans **piVars[]**. Par conséquent, ce lien ne crée jamais une valeur **cHash**.

```

// insert feinfo
$info = t3lib_div::makeInstance('tx_ccfeinfo');
$info->init($this);
$content.= $info->pi_getInfoOutput();

return $this->pi_wrapInBaseClass($content);
}

```

Pour finir, **tx_ccfeinfo** est intégré afin d'afficher les variables GET- et POST.

Ce plugin, de par les fonctions qu'il contient, est d'une grande utilité pour effectuer des tests ou pour afficher des données.

7.6 Programmation frontend : exemples

Les exemples de programmation ont été développés sous la version 3.7 de TYPO3 et installés sous la version 3.8. Certaines parties du code ont été laissées de côté pour des raisons de clarté, telles que les balises `<?php ?>` et la ligne suivante que l'on retrouve dans les fichiers comme `ext_tables.php` et `ext_localconf.php` :

```
if (!defined ('TYPO3_MODE'))      die ('Access denied.');
```

Mais tous ces éléments sont créés par le Kickstarter, de telle sorte que les exemples peuvent être reconstruits sans problèmes. De plus, tous les exemples peuvent être téléchargés.

Le préfixe `user_`, qui est réservé aux extensions locales, est toujours inséré dans les clés d'extension.

Les exemples ont été développés sur base de l'extension `CSS Styled Content`, qui restitue le contenu uniquement au travers de classes CSS. Ce concept est aussi utilisé dans les exemples d'extension. L'avantage est que la forme est plus facilement séparée du contenu. La plupart des exemples sont fonctionnels sans adaptation du gabarit standard `content (default)` ; mais il est possible que l'affichage du résultat diffère quelque peu. Cependant, ce n'est pas significatif dans le cadre du développement d'un plugin.

7.6.1 Bordures d'éléments de contenu

Cet exemple est une petite extension qui insère de nouvelles bordures pour les éléments de contenu.

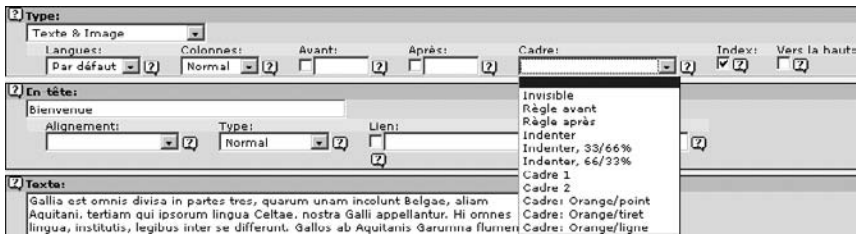


Figure 7.31:
Élément de contenu
avec de nouvelles
bordures

Pour créer de nouvelles bordures, vous devez bien sûr connaître les fonctions existantes. Puisque ces dernières ne sont pas regroupées au sein d'une extension où on peut les retrouver facilement, vous devez faire une petite recherche. Comme vous le savez, les éléments de contenu sont enregistrés dans la table `tt_content` de la base de données, et sont ensuite restitués par la configuration TypoScript `tt_content`. Il y a donc plusieurs endroits où vous pouvez commencer votre recherche sur les bordures.

Le fichier `typo3/sysex/cms/tbl_tt_content.php` contient le tableau de définition TCA pour les champs de la table `tt_content`. Malheureusement, à première vue, cette définition n'indique rien qui pourrait vous aider. Comme décrit à la section 7.4.7, l'affichage des formulaires du backend des éléments de contenu est défini dans ce fichier. Par conséquent, les données pour la sélection des bordures devraient aussi s'y trouver. Ces données, cependant, ont été déplacées dans un fichier séparé, qui contient les différentes langues. En examinant le fichier

tbl_tt_content.php, vous remarquez que les noms des champs se trouvent dans le fichier lo-callang_ttc.php⁸. On retrouve rapidement les entrées pour le champ de sélection. Les clés pour les noms des champs sont connues.

```
'section_frame' => 'Cadre:',
'section_frame.I.1' => 'Invisible',
'section_frame.I.2' => 'Règle avant',
'section_frame.I.3' => 'Règle après',
'section_frame.I.4' => 'Indenter',
'section_frame.I.5' => 'Identer, 33/66%',
'section_frame.I.6' => 'Identer, 66/33%',
'section_frame.I.7' => 'Cadre 1',
'section_frame.I.8' => 'Cadre 2',
```

En utilisant la clé (`section_frame`) dans les définitions du TCA, vous pouvez à présent rechercher le champ pour les bordures (c'est aussi possible via **Outils** → **Configuration** → **\$TCA**). La section suivante se trouve dans le fichier `tbl_tt_content.php` :

```
'section_frame' => Array (
    'exclude' => 1,
    'label' => 'LLL:EXT:cms/locallang_ttc.php:section_frame',
    'config' => Array (
        'type' => 'select',
        'items' => Array (
            Array('', '0'),
            Array('LLL:EXT:cms/locallang_ttc.php:section_frame.I.1', '1'),
            Array('LLL:EXT:cms/locallang_ttc.php:section_frame.I.2', '5'),
```

section `frame` dans la table `tt_content` définit ici les bordures pour les éléments de contenu. À présent, il faut trouver l'endroit dans la configuration TypeScript qui restitue le contenu ; à cet effet, utilisez le *TypeScript Object Browser* dans le module **Web** → **Gabarit**.

Dès lors, vous connaissez la position dans la configuration TypoScript où les nouvelles bordures doivent être insérées.

Figure 7.32:
Recherche dans la
configuration
TypeScript



⁸La version 3.8 reprend probablement les fichiers locallang dans un format XML : locallang_ttc.xml.

Pour afficher de nouvelles bordures, une nouvelle entrée est donc nécessaire dans l'objet `TypoScript` :

```
tt_content.stdWrap.innerWrap.cObject
```

ainsi que dans

```
$TCA['tt_content']['columns']['section_frame']['config']['items']
```

pour que les bordures soient disponibles pour l'utilisateur dans le backend.

Malheureusement, le Kickstarter ne prévoit pas la création d'une extension de ce type. Vous pouvez copier les fichiers nécessaires à partir d'une autre extension et les adapter, ou vous pouvez créer une extension à l'aide du Kickstarter qui contient l'essentiel de ce dont vous avez besoin, et ensuite apporter les modifications en supprimant les composants superflus.

Pour cette extension, il est recommandé de créer un plugin **Add as a 'Textbox' type** à partir du Kickstarter. Puisqu'un plugin PHP n'est pas indispensable ici, le répertoire `pi1/` peut être supprimé. L'extension contient alors les fichiers suivants :

```
ext_emconf.php
ext_icon.gif
ext_localconf.php
ext_tables.php
locallang_db.php
```

L'extension des définitions du TCA s'opère dans le fichier `ext_tables.php` et ressemble à ceci :

```
// add new frames to select box
t3lib_div::loadTCA('tt_content');

for ($key = 75; $key <= 77; $key++) {
    $TCA['tt_content']['columns']['section_frame']['config']['items'][] =
        Array('LLL:EXT:'.$_EXTKEY.
            '/locallang_db.php:tt_content.section_frames_'.$key, $key);
}
```

La définition TCA doit d'abord être chargée à partir de la table `tt_content` par la fonction `t3lib_div::loadTCA()`, afin d'être étendue en utilisant une boucle `for()`. Cette façon de procéder n'est pas indispensable, mais rend le code plus facilement réutilisable, puisque seules les valeurs de clé de départ (75) et de fin (77) doivent être adaptées.

La définition correspondante des noms de champs se trouve dans `locallang_db.php`.

```
$LOCAL_LANG = Array (
    'default' => Array (
        'tt_content.section_frames_75' => 'Frame: orange/dotted',
        'tt_content.section_frames_76' => 'Frame: orange/dashed',
        'tt_content.section_frames_77' => 'Frame: orange/solid',
    ),
    'de' => Array (
        'tt_content.section_frames_75' => 'Rahmen: Orange/gedunktet',
        'tt_content.section_frames_76' => 'Rahmen: Orange/gestrichelt',
```

```

        'tt_content.section_frames_77' => 'Rahmen: Orange/durchgezogen',
    ),
    'fr' => Array (
        'tt_content.section_frames_75' => 'Cadre: Orange/point',
        'tt_content.section_frames_76' => 'Cadre: Orange/tiret',
        'tt_content.section_frames_77' => 'Cadre: Orange/ligne',
    ),
);

```

Une enveloppe est nécessaire pour créer la bordure. **Border 1** peut servir ici de modèle :

```

tt_content.stdWrap.innerWrap.cObject.20 = TEXT
tt_content.stdWrap.innerWrap.cObject.20.value = <div class="csc-frame cs
c-frame-frame1">|</div>

```

Le code TypeScript est inséré dans `ext_localconf.php`. On recourt aussi ici à une boucle `for` pour générer le TypeScript nécessaire via `t3lib_extMgm::addTypoScript()`.

```

// generate frames with key 75 - 77
for ($key = 75; $key <= 77; $key++) {
    t3lib_extMgm::addTypoScript($_EXTKEY, 'setup', '
tt_content.stdWrap.innerWrap.cObject.' . $key . ' = TEXT
tt_content.stdWrap.innerWrap.cObject.' . $key . '
    ' . value = <div class="csc-frame-frame' . $key . '">|</div>
    ', 43);
}

```

Ce qui manque encore, c'est un peu de CSS pour rendre les bordures visibles. On pourrait l'inclure dans la feuille de style du site Web. Mais vous pouvez aussi l'insérer si nécessaire via le gabarit de l'extension. Pour ce faire, créez un nouveau fichier :

```

static/
    setup.txt

```

Ce fichier doit contenir du code qui ressemble à ceci (en abrégé) :

```

# Example of default set CSS styles (these go into the document header):
plugin.tx_userframes._CSS_DEFAULT_STYLE (
DIV.csc-frame-frame75 {
    background-color: #FAAC27; border: 3px dotted #000; }
DIV.csc-frame-frame76 {
    background-color: #FAAC27; border: 3px dashed #000; }
DIV.csc-frame-frame77 {
    background-color: #FAAC27; border: 3px solid #000; }
)

```

L'extension est à présent terminée et peut être installée par le gestionnaire d'extensions. Pour afficher les bordures, le CSS fourni peut être intégré en ajoutant le gabarit statique à l'enregistrement de gabarit.

Figure 7.33:
Ajout du gabarit
statique



Un des désavantages de ce procédé est que le code CSS est inséré dans l'en-tête de la page HTML, ce qui rend celle-ci plus lourde. D'un autre côté, on peut agir de la sorte délibérément, pour associer le code CSS à une page en particulier, évitant ainsi le chargement d'une feuille de style externe.

La figure 7.34 suivante vous présente le résultat de cette petite extension.

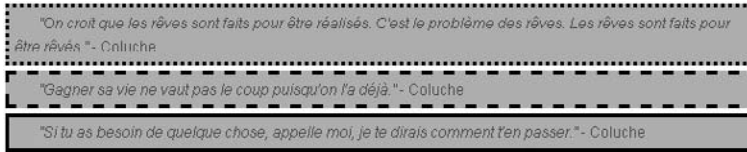


Figure 7.34:
Trois nouveaux types
de bordure dans le
frontend

Le code généré a l'allure suivante :

```
<style type="text/css">
  /*<![CDATA[*/
<!--
/* default styles for extension "tx_userframes" */
DIV.csc-frame-frame75 {
  background-color: #FAAC27; border: 3px dotted #000; }
DIV.csc-frame-frame76 {
  background-color: #FAAC27; border: 3px dashed #000; }
DIV.csc-frame-frame77 {
  background-color: #FAAC27; border: 3px solid #000; }
-->
  /*]]>*/
</style>
...
</head>
<body>
...
<!-- CONTENT ELEMENT, uid:15/text [begin] -->
  <a name="15"></a><div class="csc-frame-frame75">
    <!-- Text: [begin] -->
    <p class="bodytext">On croit que les rêves ...</p>
    <!-- Text: [end] -->
  </div>
<!-- CONTENT ELEMENT, uid:15/text [end] -->
```

7.6.2 La balise Typo de compte à rebours (TypoTag)

Bien qu'un CMS doive séparer la forme du contenu, il est souvent nécessaire de marquer le contenu au préalable. TYPO3 fournit ses propres balises pour ce marquage. Il ne faut pas confondre ces dernières avec les balises HTML, car les *balises Typo* sont transformées en balises HTML (ou un autre format) durant le processus de restitution.

L'exemple suivant introduit la balise `<countdown>`. Son but est de décompter les jours jusqu'à une certaine date. L'élément **Text** contenant :

Il reste `<countdown>29 Jan 2006</countdown>` jours avant le TYPO3 Snowboard Tour 2006.

doit donner le résultat suivant :

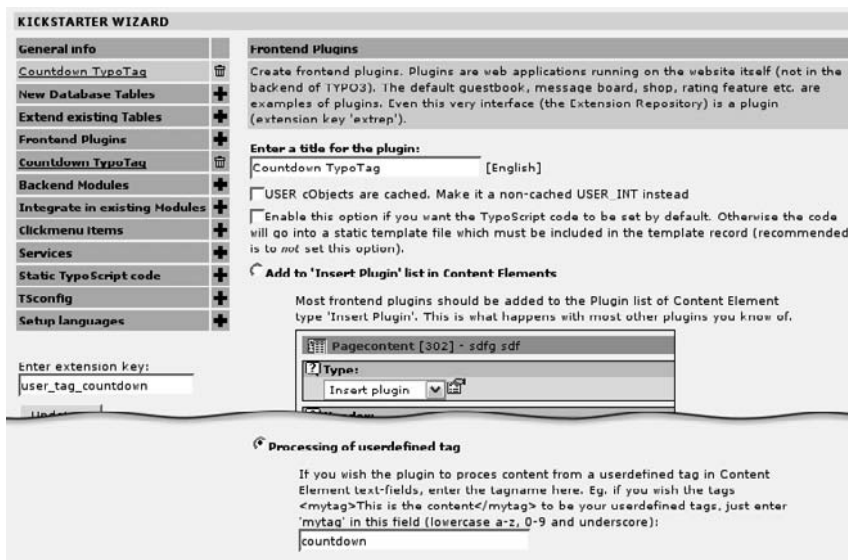
Il reste 103 jours avant le TYPO3 Snowboard Tour 2006.

La date contenue dans la balise doit donc être convertie en nombre de jours restant avant cette date.

Une nouvelle balise offrant cette fonctionnalité peut facilement être créée dans le Kickstarter. Le Kickstarter fournit les fichiers suivants :

```
ext_emconf.php
ext_icon.gif
ext_localconf.php
pi1/
    class.user_tagcountdown_pi1.php
```

Figure 7.35:
Balise <countdown>
créée en tant que
plugin dans le
Kickstarter



Dans le fichier `ext_localconf.php`, le script PHP `class.user_tagcountdown_pi1.php` et le plugin sont intégrés dans le gabarit TypoScript.

```
t3lib_extMgm::addPIToST43($_EXTKEY,
    'pi1/class.user_tagcountdown_pi1.php',
    '_pi1',
    '',
    1);

t3lib_extMgm::addTypoScript($_EXTKEY, 'setup', '
lib.parseFunc_RTE.tags.countdown = < plugin.'.
t3lib_extMgm::getCN($_EXTKEY).
    '_pi1
', 43);
```

Remarquez ici que cela ne fonctionne pas avec le gabarit content (default), car `parseFunc` se trouve à un autre endroit. Si le plugin doit opérer avec ce gabarit, il doit être intégré d'une autre façon. Généralement, le TypeScript Object Browser dans le module **Web** → **Gabarit** nous donne de l'information à ce propos.

```
t3lib_extMgm::addTypoScript($_EXTKEY, 'setup', '
    tt_content.text.20.parseFunc.tags.countdown = < plugin.'.
    t3lib_extMgm::getCN($_EXTKEY).
    '_pil
', 43);
```

Mais revenons à la première situation. Voici le résultat, en résumé de l'intégration du TypoScript :

```
plugin.user_tagcountdown_pil = USER
plugin.user_tagcountdown_pil.userFunc = user_tagcountdown_pil->main

includeLibs.user_tagcountdown_pil = typo3conf/ext/user_tag_countdown/pil
/class.user_tagcountdown_pil.php

lib.parseFunc_RTE.tags.countdown = < plugin.user_tagcountdown_pil
```

Vous pouvez aussi spécifier ceci directement en tant que gabarit TypoScript dans le fichier `ext_setup.txt`. Mais la meilleure façon de procéder, qui est aussi la plus portable, est d'intégrer le plugin via `t3lib_extMgm`, puisque aucun chemin de fichiers statiques n'est utilisé.

Le script final `class.user_tagcountdown_pil.php` se présente alors comme suit :

```
require_once(PATH_t3lib.'class.t3lib_pibase.php');

class user_tagcountdown_pil extends t3lib_pibase {
    var $prefixId = 'user_tagcountdown_pil';
    var $scriptRelPath = 'pil/class.user_tagcountdown_pil.php';
    var $extKey = 'user_tag_countdown';

    /**
     * processes the <countdown> tag
     */
    function main($content, $conf) {
        $date = $this->cObj->getCurrentVal();
        $timestamp = strtotime($date);
        $delta = $timestamp - time();
        $days = intval($delta / (60 * 60 * 24)) + 1;

        return $days;
    }
}
```

La fonction `$this->cObj->getCurrentVal()` renvoie le contenu de la balise `<countdown>` (la date), qui est convertie en jours dans les lignes suivantes. Remarquez que la fonction `strtotime()` ne peut pas manipuler les conventions internationales, mais cela n'a pas d'importance dans cet exemple.

Les paramètres sont passés à la fonction `main()` de deux manières. Pour la première, le passage se fait directement dans la balise Typo, comme pour les balises HTML.

Il reste <countdown unit=hour>29 Jan 2006</countdown> heures avant le TYPO3 Snowboard Tour 2006.

Tous les paramètres dans la balise sont accessibles par l'objet plugin via le tableau `$this->cObj->parameters`. Dans ce cas, `$this->cObj->parameters['unit']` contient l'heure. On peut aussi travailler de la sorte avec plusieurs paramètres. De plus, `$this->cObj->parameters['allParams']` contient tous les paramètres sous forme de chaînes de caractères, dans le format dans lequel ils sont passés. Pour afficher les heures, on doit apporter les modifications suivantes dans le plugin :

```
function main($content,$conf) {
    // get parameter from tag
    $unit = $this->cObj->parameters['unit'];
    // get parameter from TypoScript
    $unit = $unit ? $unit : $this->cObj->stdWrap($conf['unit'], $conf['unit.']);

    if ($unit=='min') {
        $divider = 60;
    } elseif ($unit=='hour') {
        $divider = 60 * 60;
    } else {
        // day - default
        $divider = 60 * 60 * 24;
    }

    $date = $this->cObj->getCurrentVal();
    $timestamp = strtotime($date);
    $delta = $timestamp - time();
    $days = intval($delta / $divider) + 1;

    return $days;
}
```

À présent, la balise <countdown> peut gérer des minutes, des heures et des jours.

La deuxième manière de passer les paramètres, comme pour tout plugin, est de recourir au TypoScript. Le code qui récupère le paramètre `unit` est déjà présent dans le code ci-dessus :

```
$unit = $unit ? $unit : $this->cObj->stdWrap($conf['unit'], $conf['unit.']);
```

Si `unit` n'a pas été passé en paramètre dans la balise, la fonction `stdWrap()` est appelée pour récupérer la valeur via la configuration TypoScript. `stdWrap()` est un composant de l'objet de contenu `cObj`. L'objet `cObj` est instancié de manière externe lorsque le plugin est initialisé, et reste à la disposition du plugin. `cObj`, une instance de `tslib/class.tslib_content.php`, restitue tous les objets connus de TypoScript tels que `TEXT` ou `IMAGE`.

```
plugin.user_tagcountdown_pi1.unit = min
```

Cette configuration TypoScript spécifie que la valeur de `unit` est `min`. La valeur par défaut n'est donc plus `day`. Donc, les minutes sont décomptées, aussi longtemps qu'une autre valeur de `unit` n'est pas spécifiée dans la balise <countdown>.

Puisque `plugin.user_tagcountdown_pi1.unit` est un objet de type `stdWrap`, il hérite des mêmes propriétés. La configuration TypoScript suivante est dès lors possible :

```
plugin.user_tagcountdown_pi1.unit = min
plugin.user_tagcountdown_pi1.unit.override.data = register: user_tagcountdown_unit
```

Ici, `unit` reçoit la valeur `min`, mais si `user_tagcountdown_unit` contient déjà une valeur, alors cette dernière est remplacée par le registre TS. TYPO3 lit le TypoScript dans un tableau PHP. Vous pouvez afficher ce tableau en insérant la fonction `debug($conf)` au début. Si vous rechargez maintenant la page dans le frontend, vous obtiendrez le résultat présenté à la figure 7.36.

userFunc	user_tagcountdown_pi1->main		
unit	min		
unit.	override.	data	register: user_tagcountdown_unit

Figure 7.36:
Affichage par la
fonction `debug()`

Le TypoScript complet pour le plugin est affiché. Les tables reproduisent l'imbrication des tableaux. Sous forme de texte, voici ce que cela donne :

```
plugin.user_tagcountdown_pi1 = USER
plugin.user_tagcountdown_pi1 {
    userFunc = user_tagcountdown_pi1->main
    unit = min
    unit.override.data = register: user_tagcountdown_unit
}
```

Si `stdWrap()` est appelé, le premier argument de la fonction est la valeur de `$conf['unit']`, dans ce cas, `min`.

```
$this->obj->stdWrap($conf['unit'], $conf['unit.']);
```

Le premier argument de `stdWrap()` est la valeur ou le contenu que la fonction est censée traiter. Le second argument `$conf['unit.']` définit le comportement de `stdWrap()` au moyen de paramètres contenus dans un tableau. Vous devriez étudier la fonction `stdWrap()` dans `tslib/class.tslib_content.php` et essayer de comprendre son fonctionnement. Vous retrouverez ce concept encore et toujours dans TYPO3. Et si vous réexaminez le code PHP de ce plugin, vous réaliserez que même la fonction `main()` utilise ce concept.

7.6.3 Balise de compte à rebours en JavaScript

La balise de compte à rebours sera ici étendue pour que le compteur dans le frontend continue à décompter via JavaScript. Pour que cela ait du sens, nous devons ici étendre le plugin pour qu'il affiche les secondes. De plus, le paramètre `animate` est introduit.

```
Il reste <countdown unit=sec animate>29 Jan 2006</countdown> secondes avant le
TYPO3 Snowboard Tour 2006.
```

Le code JavaScript est intégré dans la page uniquement au cas où le compte à rebours doit être affiché. Même si cela sort du cadre strict de ce livre, nous examinons ici brièvement la

programmation JavaScript. Afin de différencier le code JavaScript inclus par plusieurs extensions dans le code HTML des pages, le JavaScript doit tenir compte de l'espace de nommage des extensions. Le moyen le plus simple est d'encapsuler l'entièreté du script dans un objet, ce qui évite de recourir à des variables globales.

Les objets JavaScript jouent le même rôle que les classes en PHP. Une fonction sert ici de conteneur qui peut être étendu par la propriété **prototype**.

```
function myClass() {
    this.myInstanceVar = 0;
}

myClass.prototype.myFirstMethod = function (param1, param2) {
    this.myInstanceVar = (param1 + param2)/2;
}

myClass.prototype.mySecondMethod = function () {
    return this.myInstanceVar;
}
```

De cette façon, seule la fonction **myClass()** est présente dans l'espace de nommage global. Toutes les autres fonctions et variables sont encapsulées. Pour ce faire, notre choix se porte naturellement sur **\$this->extKey** ou **\$this->prefixId** comme espace de nommage pour le code JavaScript. Voici donc la structure du JavaScript pour l'extension :

```
function user_tagcountdown_p11(id,countdown,unit) {
    ...
}

user_tagcountdown_p11.prototype.showcount = function () {
    ...
}
```

Le tableau **additionalHeaderData** de l'objet global **TSFE** est à votre disposition pour insérer du code dans l'en-tête HTML. L'exemple suivant vous fournit un exemple d'intégration du JavaScript dans l'en-tête.

```
// check if JavaScript is already set
if (!$GLOBALS['TSFE']->additionalHeaderData[$this->prefixId]) {

    $jsCode = '
        alert("This is JavaScript");';

    // wrap JavaScript in script tags and add to page header
    $GLOBALS['TSFE']->additionalHeaderData[$this->prefixId] =
        t3lib_div::wrapJS($jsCode);
}
```

On vérifie d'abord si du code JavaScript a déjà été inséré. En effet, ce code doit apparaître une seule fois, quel que soit le nombre d'appels à ce plugin. Grâce à la fonction **t3lib_div::wrapJS()**, le code JavaScript est entouré par les balises **<script>** avant d'être ajouté dans l'en-tête HTML.

Si vous voulez utiliser le traitement d'événement JavaScript `onload` dans la balise `body`, vous devez bien sûr tenir compte des autres extensions. À cette fin, entrez le code JavaScript `onload` dans un tableau de l'objet `TSFE`.

```
$GLOBALS['TSFE']->JSEventFuncCalls['onload'][$this->prefixId] =
    'alert("JS executed on load");';
```

Dans le code modifié du plugin, un compteur est d'abord introduit pour donner un identifiant unique (ID) aux éléments. On utilise pour ce faire une variable globale :

```
function main($content,$conf) {
    // count items for JavaScript usage
    $GLOBALS['T3_VAR']['ext'][$this->prefixId]['count']++;
```

Grâce à `$GLOBALS['T3_VAR']['ext'][$this->prefixId]`, seul l'espace de nommage du plugin est utilisé.

```
    // get parameter from tag
    $unit = $this->cObj->parameters['unit'];
    // get parameter from TypeScript
    $unit = $unit ? $unit : $this->cObj->stdWrap($conf['unit'],
        $conf['unit.']);
```

Les secondes sont alors ajoutées dans le décompte :

```
    if ($unit=='sec') {
        $divider = 1;
    } elseif ($unit=='min') {
    ...
    $days = intval($delta / $divider) + 1;
```

Le nouveau paramètre `animate` est inséré. Notez que la fonction `isset()` vérifie que `animate` est bien présent, car vous devez le spécifier dans la balise comme suit : `<countdown animate=1 ...>`.

```
    // check if "animate" parameter is set in tag
    $animate = isset($this->cObj->parameters['animate']);
    // get parameter from TypeScript
    $animate = $animate ? $animate :
        $this->cObj->stdWrap($conf['animate'], $conf['animate.']);
```

Pour finir, voici le code qui intègre l'animation. Celle-ci tient compte des secondes ou des minutes, puisque personne ne noterait de changement sur les heures ou les jours.

```
    // do animation for seconds or minutes only
    if ($animate AND ($unit=='sec' OR $unit=='min')) {
```

Un ID pour l'élément HTML est créé :

```
    // unique id for every element (HTML DOM doesn't accept '_' )
    $domId = str_replace('_', '-', $this->prefixId).'-'.
        $GLOBALS[$this->prefixId]['count'];
```

On appelle la fonction, qui est présentée plus tard, pour créer le code JavaScript.

```
// include JS code
$this->addJsCounter($domId, $days, $unit);
```

Pour finir, le texte est entouré par les balises `<span>` incluant un ID.

```
// add an id to the content
$days = '<span id="' . $domId . '">' . $days . '</span>';
}

return $days;
}
```

Vient à présent la fonction qui insère le code JavaScript. Les endroits contenant du JavaScript sont en gras dans le texte. Notez que le code PHP comprend des variables telles que `{ $this->prefixId }` qui rendent le code portable.

```
/**
 * Add JavaScript counter code to the page
 * @param string dom id
 * @param integer start value for the counter
 * @param string unit: 'sec' or 'min'
 * @return void
 */
function addJsCounter($id, $countdown, $unit) {
    // include JS code once
    if (!$GLOBALS['TSFE']->additionalHeaderData[$this->prefixId]) {

        $jsCode = <<<EOD
        {$this->prefixId}ObjArr = new Array();

        function {$this->prefixId}(id,countdown,unit) {
            this.id = id;
            this.countdown = countdown;
            setInterval(
                "{ $this->prefixId }ObjArr['"+id+"'].showcount()",
                unit);
        }

        {$this->prefixId}.prototype.showcount = function () {
            this.countdown = this.countdown-1;
            element = document.getElementById(this.id);
            element.innerHTML = this.countdown;
        }

        EOD;

        // wrap JavaScript in script tags and add to page header
        $GLOBALS['TSFE']->additionalHeaderData[$this->prefixId] =
            t3lib_div::wrapJS($jsCode);
    }

    $unit = ($unit=="min") ? 60000 : 1000;

    // add JS onload handler
    $GLOBALS['TSFE']->JSEventFuncCalls['onload'][$id] =
```

```

    "{$this->prefixId}ObjArr['{$id}'] =
      new {$this->prefixId}('{$id}', {$countdown}, '{$sunit}');";
  }

```

Puisque les principes ont déjà été exposés, nous ne reviendrons pas en détail sur le code de l'exemple.

Comme vous le constatez, il n'est pas si difficile d'intégrer du JavaScript pour que plusieurs plugins soient actifs sur la même page, sans entrer en conflit les uns avec les autres.

7.6.4 Intégration de scripts PHP externes

Au plus les scripts sont simples et bien structurés, au plus il sera facile de les intégrer dans TYPO3. Si les scripts sont déjà organisés en classes, l'écriture d'un « wrapper »⁹ semble la solution la plus évidente à mettre en place. Ce dernier intègre la classe et l'appelle à bon escient. Plus de travail est nécessaire si des paramètres sont transférés. En règle générale, l'URL et, si nécessaire, les paramètres eux-mêmes doivent être adaptés. Si le script est peu structuré ou si c'est un script ancien qui utilise des variables PHP maintenant obsolètes, des adaptations doivent être apportées.

Référence 680352

Des scripts PHP externes peuvent être intégrés de différentes façons dans le frontend, via les objets TypoScript suivants :

USER, USER_INT

Ces objets représentent la méthode standard. Tous les plugins créés par le Kickstarter sont intégrés en tant qu'objets **USER** ou **USER_INT**. La différence entre les deux méthodes repose sur le fait que le résultat est caché lors de l'intégration par **USER**, alors que ce n'est pas le cas avec **USER_INT**. Cette dernière méthode est utilisée si le script doit tenir compte des paramètres (GET, POST) lors de l'affichage du résultat, et lorsque cela ne vaut pas la peine de recourir à la fonctionnalité **cHash**. L'intégration par **PHP_SCRIPT** est conseillée uniquement si un script n'est pas encapsulé dans une classe, et si le portage n'est pas envisageable.

PHP_SCRIPT

L'intégration d'un script ressemble à ceci dans TypoScript :

```

page.90 = PHP_SCRIPT
page.90.file fileadmin/scripts/myscript.php

```

Le script est inclus dans **tslib_cObj**, ce qui signifie que vous avez accès à toutes les méthodes de cet objet via **\$this**. Le résultat du script doit être contenu dans la variable **\$content**.

PHP_SCRIPT_INT

Cette méthode se comporte de la même manière que **PHP_SCRIPT**, à ceci près que le résultat du script n'est pas caché.

PHP_SCRIPT_EXT

Avec cette méthode, le résultat n'est pas caché. Mais l'intégration est différente puisque le résultat peut être affiché par les commandes **echo/print**.

⁹NdT : littéralement, programme enveloppant

Convertir un script PHP

Si vous voulez intégrer des scripts dans TYPO3, alors qu'ils n'ont pas été développés dans ce but, c'est généralement possible sans trop d'efforts. Nous illustrons ci-après les différentes possibilités existantes, sur base d'un petit script qui extrait quelques informations sur un serveur Web.

Figure 7.37:
Affichage par le script
"Examine server"

Examine Server

Enter a domain name to examine their server:

Go

typo3.com is running:

Server: Apache/1.3.28 (Unix) PHP/4.2.2
X-Powered-By: PHP/4.2.2

Voici la version originale du script :

```
<?php
# based on a script from http://px.sklar.com by Matt DeLong

// examine server information with GET request
function examine($domain){
    $result = '';

    if($domain){
        if($fp = @fsockopen($domain, 80, &$errno, &$errstr, 30)){
            fputs($fp, "GET / HTTP/1.0\r\n\r\n");
            $data = array();
            while(!feof($fp)) {
                $data[] = fgets($fp, 128);
            }
            fclose ($fp);
        }

        for($x=0; $x<7; $x++) {
            $result = (strstr(strtolower($data[$x]), 'server:')) ?
                                                                $data[$x] : $result;
        }
        $result.= (strstr($data[3], 'X-Powered-By')) ?
                                                                '<br \>'. $data[3] : '';
    }

    $result = ($domain && !$result) ?
        '<b>ERROR:</b> connection could not be established with '.
        htmlspecialchars($domain) : $result;
    return $result;
}
$domain = stripslashes($HTTP_POST_VARS['domain']);

?>
<html>
```

```

<body>
<h2>Examine Server</h2>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
  <p>Enter a domain name to examine their server:
  <input type="text" name="domain" value="<?php
    echo htmlspecialchars($domain); ?>">
  <input type="submit" value="Go"></p>
</form>
<hr>
<p>
<?php
  $result = examine($domain);
  if($result && !strstr($result, 'ERROR'))
    echo htmlspecialchars($domain).' is running:<br><br>';
  echo $result;
?>
</p>
</body>
</html>

```

Il est possible d'obtenir une page complète à partir de ce script. La configuration TypoScript suivante fait simplement appel au script ci-dessus et ne crée aucun autre résultat :

```

page >
page = PAGE
page.typeNum = 0
page.config.disableAllHeaderCode = 1

page.50 = PHP_SCRIPT_EXT
page.50.file = fileadmin/scripts/examine-server-ext.php

```

Mais si vous intégrez le script, comme dans les deux dernières lignes, dans votre propre page (avec votre propre configuration de `page`), il faut supprimer du script tout ce qui a déjà été généré par TYPO3. Normalement, il s'agit de l'en-tête HTML et de la balise `<body>`. De plus, par opposition à `PHP_SCRIPT_EXT`, les commandes `echo/print` ne sont pas possibles pour `PHP_SCRIPT` et `PHP_SCRIPT_INT`. Ici, le script doit renvoyer le contenu dans la variable `$content`. On réalise ceci facilement via la bufferisation de sortie de PHP.

```

...
ob_start();

?>
<h2>Examine Server</h2>
...

<?php

$content = ob_get_contents();
ob_end_clean();

?>

```

Un problème fréquent lors de l'intégration de scripts est le traitement des paramètres et des URL. Aussi, ce script ne fonctionne pas, car l'URL de destination du formulaire n'est pas correcte. La version corrigée du script est affichée ci-dessous dans sa variante `PHP_SCRIPT_INT`.

```
<?php
# based on a script from http://px.sklar.com by Matt DeLong

function examine($domain){
    ...
    return $result;
}

$domain = tslib_div::_GP('domain');

ob_start();
?>
<h2>Examine Server</h2>
<form action="<?php
    echo htmlspecialchars(tslib_div::getIndpEnv('REQUEST_URI'));
?>" method="POST">
    <p>Enter a domain name to examine their server:
    <input type="text" name="domain" value="<?php
        echo htmlspecialchars($domain); ?>"
    <input type="submit" value="Go"></p>
</form>
<hr>
<p>
<?php
    $result = examine($domain);
    if($result && !strstr($result, 'ERROR'))
        htmlspecialchars($domain).' is running:<br><br>';
    echo $result;
?>
</p>
<?php

$content = ob_get_contents();

ob_end_clean();
?>
```

La question qui se pose est de savoir s'il ne serait pas préférable de procéder via une extension et de créer un plugin. En effet, cela requiert à peine plus de travail, un meilleur cadre est établi pour les futurs développements et le script est intégré correctement. De plus, l'intégration via `PHP_SCRIPT` peut conduire à des instabilités dans TYPO3 si le script manipule des variables globales. La transformation en plugin ressemble à ceci :

```
class user_exmsv_pil extends tslib_pibase {

    var $prefixId = 'user_exmsv_pil';
    var $scriptRelPath = 'pil/class.user_exmsv_pil.php';
    var $extKey = 'user_exm_sv';
```

```

/**
 * Examine Server main function
 */
function main($content,$conf) {
    $this->conf=$conf;
    $this->pi_setPiVarDefaults();
    $this->pi_USER_INT_obj=1;

    $domain = $this->piVars['domain'];

    $result = $this->examine($domain);

    if($result && !strstr($result, 'ERROR')) {
        $result = '
            <hr>
            <p>'.htmlspecialchars($domain).' is running:<br />
            <br />
            ' . $result . '</p>';
    }

    $content = '
        <h2>Examine Server</h2>
        <form action="'.
            htmlspecialchars(t3lib_div::getIndpEnv('REQUEST_URI')).
            '" method="post">

            <p>Enter a domain name to examine their server:
            <input type="text" name="'. $this->prefixId.
                '[domain]' value="'. htmlspecialchars($domain) .'">
            <input type="hidden" name="no_cache" value="1" />
            <input type="submit" value="Go"></p>
        </form>' . $result;

    return $this->pi_wrapInBaseClass($content);
}

/**
 * Examine server information with GET request
 *
 * @param    string    domain name
 * @return    string
 */
function examine($domain){
    ...
    return $result;
}
}

```

Le principal changement consiste à utiliser les paramètres système au travers de la classe `tslib_piBase`. Par conséquent, tous les paramètres qui sont passés avec `$this->prefixId` en tant qu'éléments d'un tableau sont automatiquement disponibles dans `$this->piVars[]`.

```
$domain = $this->piVars['domain'];

<input type="text" name="'. $this->prefixId.'[domain]' value="'.
    htmlspecialchars($domain).' ">
```

Pour un petit script tel que celui-ci, cela vaut la peine de le transformer en plugin. La fonctionnalité du script est suffisamment claire, ce qui le rend également facile à réutiliser.

7.6.5 Portage de script PHP

En utilisant l'exemple de l'application PHP *dataMiner*¹⁰, nous allons montrer comment porter des scripts vers TYPO3. *dataMiner* est utilisé pour afficher des tables de base de données sous forme de listes et de vues détaillées. Des enregistrements peuvent aussi être édités individuellement.

Figure 7.38:
Affichage de liste par
dataMiner

Name	Title	Address	Phone	Fax	Email	Company	City	Zipcode	Country	Description
Abraham "Stranger" Simpson		Springfield Retirement Castle			stranger@springfield.net		Springfield	1234	USA	
Ville	Groundkeeper					Springfield Elementary School	Springfield	1234	USA	
Chief Higgins	Chief of Police				higgins@springfield.net		Springfield	1234	USA	
Max Scyzalk	Defender	Moore Tavern	(987) 654 321	(987) Name	max@springfield.net	Moore Tavern	Springfield	1234	USA	
Marge Simpson		Evergreen Terrace 473			marge@springfield.net		Springfield	1234	USA	
J. Doe TEST (DUM, Funt, Books)		His address	+1 808-2020 202		jdoe.doe@test.net		What Ever Town	1234		
Jane Doe TEST (PLANK, Sport, Pigeons)		His address	+1 808-2020 202		jane.doe@test.net		What Ever Town	1234		
Sally Silvermoon		Her address in some state	+1 808-2020 202		sally@silvermoon.net		What Ever Town	1234		
Abraham "Stranger" Simpson		Springfield Retirement Castle			stranger@springfield.net		Springfield	1234	USA	
Ville	Groundkeeper					Springfield Elementary School	Springfield	1234	USA	

Le but du portage est d'intégrer autant que possible *dataMiner* dans TYPO3, tout en minimisant l'effort nécessaire.

Le répertoire *dataMiner* contient les fichiers suivants :

```
dataMiner-0.20.0/
  LICENSE
  NOTES
  README
  img/
    asc.gif
    cal.gif
    delete.gif
    ...
  index.html
  index.php
  lib/
    dm.js
    dataBrowser.php
    dataDBI.php
    dataDetailer.php
    dataEditor.php
    dataMiner.php
```

¹⁰<http://greenhell.com/dataMiner>

```
metabase/
...
sample.sql
```

Le code source de dataMiner est programmé de manière très structurée. Les fichiers du répertoire `lib/` contiennent l'application. Le code est encapsulé dans des classes, ce qui devrait faciliter le portage.

L'application doit être intégrée plus tard dans une page en tant que plugin. Un plugin sans cache de type (`USER_INT`) est créé dans le Kickstarter. Le répertoire dataMiner est alors copié vers la nouvelle extension. Les fichiers inutiles sont supprimés, mais les fichiers `LICENSE`, `NOTES` et `README` doivent être laissés là où ils sont.

Tout d'abord, le module **Outils** → **ExtDevEval** est à votre disposition pour convertir les guillemets doubles en guillemets simples (voir la section chap7 : guidelines).

Les scripts contiennent plusieurs commandes `include`. Celles-ci doivent être modifiées dans le formulaire pour que `t3lib_extMgm::extPath()` détermine le chemin de l'extension.

```
include_once('lib/dataMiner.php');

require_once(t3lib_extMgm::extPath('dataminer').'dataMiner/lib/dataMiner.php');
```

Le fichier `index.php` dans le répertoire est un exemple parmi d'autres qui illustre la façon dont les classes de l'application dataMiner sont utilisées. La structure ressemble à ceci (en abrégé) :

```
$foo = new dataBrowser();
$foo->type      = "mysql";           // database server type

$foo->name       = "MYDB";           // name of the database
$foo->user       = "root";           // database username
$foo->pass       = "";               // database Password
$foo->host       = "localhost";      // database server hostname

// The browser needs a table name and it's primary key (required)
$foo->table      = "city_directory"; // database table
$foo->key        = "Rid";            // table primary key field name

// Assigning a table title will display the title instead of the actual
// database table name.
// title = "string";
$foo->title      = "Businesses";

// print results (required)
$foo->Main();
```

Le code dans `index.php` est repris dans la méthode `main()` du plugin, dans le fichier `pi1/tx_dataminer_pi1.php`. Il faut toujours intégrer la classe principale de dataMiner dans l'en-tête du fichier.

```
require_once(t3lib_extMgm::extPath('dataminer').'dataMiner/lib/dataMiner.php');
```

Il s'avère que l'ensemble du résultat de dataMiner est généré par la méthode `Main()`. Cependant, tout le contenu est affiché par la commande `print`. Cela ne correspond bien sûr pas du

tout au concept de plugin, qui doit renvoyer le contenu via `return`. Heureusement, PHP propose des fonctions de bufferisation de résultat. La portion de code suivante montre comment la bufferisation du résultat est initiée. Ensuite, la méthode `Main()` de `dataMiner` est appelée. Le résultat issu de la commande `print` est bufferisé par PHP. Ce buffer est ensuite lu dans les lignes de code suivantes et passé à la variable `$content`. Pour finir, le buffer est désactivé et son contenu renvoyé au système.

```
ob_start();

$dm->Main();
$content = ob_get_contents();

ob_end_clean();

return $this->pi_wrapInBaseClass($content);
```

`dataMiner` utilise *Metabase* pour se connecter à la base de données. Pour pouvoir tester le plugin, la configuration suivante doit être faite pour la base de données, en utilisant les constantes de `TYPO3`.

```
$dm->type = 'mysql';           // database server type

$dm->name = TYPO3_db;          // name of the database
$dm->user = TYPO3_db_username; // database username
$dm->pass = TYPO3_db_Password; // database Password
$dm->host = TYPO3_db_host;     // database server hostname
```

C'est une solution parfaitement valide, bien que vous ne soyez pas obligé d'utiliser *Metabase*. Mais nous y reviendrons plus tard.

`dataMiner` est conçu pour fonctionner avec n'importe quel type de base de données, après un peu de configuration. Pour les besoins de notre test, la table `tt_address` est sélectionnée et configurée.

```
// The browser needs a table name and it's primary key (required)
$dm->table = 'tt_address'; // database table
$dm->key = 'uid';          // table primary key field name
```

À ce stade, le plugin peut être installé et testé. La table `tt_address` est correctement affichée sous forme de listes. Cependant, les icônes ne sont pas visibles, puisque les chemins ne sont plus valides. De plus, les liens et les formulaires ne fonctionneront plus, car les URL ne respectent pas les normes de `TYPO3`.

On adapte très simplement les chemins des icônes en utilisant les fonctions de remplacement de texte. En premier lieu, le chemin relatif à l'extension est initialisé à un point central du code de `dataMiner`.

```
function _var_setup() {
    $this->iconPath = t3lib_extMgm::siteRelPath('dataminer').'/dataMiner/';
```

Ensuite, `$this->iconPath` est inséré dans tous les chemins d'images.

```
"<img src=\"img/details.gif\" ...
```

```
"<img src=\"\".$this->iconPath.\"img/details.gif\" ...
```

Un problème supplémentaire surgit si vous examinez le code HTML du plugin dans le résultat du frontend. dataMiner produit son propre en-tête HTML, qui doit bien sûr être supprimé. La méthode `_head()` qui se trouve dans le code contient l'en-tête HTML en question.

```
function _head($title) {
    $this->_load_skin();
}
?>
<!-- $Id: dataMiner.php,v 1.11 2003/06/02 17:55:18 rlineweaver Exp $ -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title><?php echo $title; ?></title>
<meta name="robots" content="noindex">
<style>
    BODY,TD,P,H1,H2,H3,H4,FORM {
        font-family:Helvetica,Arial,sans-serif; font-size:95%; }
    .NEW {
        color:<?php echo $this->newFileColor; ?>; }
    .HDR {
        color:<?php echo $this->bodyTextColor; ?>; font-size:10pt;
        font-weight: bold; }
    .TIT {
        color:<?php echo $this->siteNameColor; ?>;
        background-color:<?php echo $this->menubarColor; ?>;
        font-size:14pt; font-weight: bold; }
    ...
</style>
</head>
<body bgcolor="#<?php echo $this->bodyBgColor; ?>"
    link="#<?php echo $this->rowLinkColor; ?>"
    ...
    marginwidth="<?php echo $this->bodyMarginSize; ?>"
<?php echo $this->_colorbars($this->headDiv); ?>

<table border=0 cellspacing=0 cellpadding=2 width="<?php
    echo $this->siteWidth; ?>">
<tr><td class="TIT"><?php echo $title; ?></td></tr>
...

```

En principe, l'entièreté de l'en-tête, y compris la balise `<body>`, peut être supprimée. La mise en forme est contrôlée par CSS, ce dernier étant lui-même défini à partir d'« habillage », que l'on voit une fois que la fonction `_load_skin()` est appelée. La fonctionnalité des habillages doit être enlevée, mais le contrôle par CSS doit demeurer. Le code CSS peut être récupéré dans l'HTML du frontend, puisque les variables ont été remplacées par les couleurs correspondantes de l'habillage par défaut. Vous pouvez ainsi supprimer à la fois l'en-tête HTML généré par `_head()` et la méthode `_load_skin()`. Dans la méthode `_foot()`, les balises `</body></html>` doivent aussi être éliminées, pour que le plugin se limite à l'affichage du contenu de dataMiner.

À présent, la page créée ne contient plus du tout de CSS, et dataMiner ne s'affiche plus correctement. Vous pourriez inclure le code CSS enregistré précédemment dans un fichier pour le site, mais il est préférable que le CSS soit associé à l'extension. C'est possible si vous utilisez TypeScript pour l'intégration de la mise en forme :

```
plugin.tx_dataminer._CSS_DEFAULT_STYLE (
.tx-dataminer-pil .NEW {
    color:ffff00; }
.tx-dataminer-pil .HDR {
    color:000000;
    font-size:10pt; font-weight: bold; }
.tx-dataminer-pil .TIT {
    color:ffffff; background-color:111111;
    font-size:14pt; font-weight: bold; }
...

```

Puisqu'une telle configuration TypeScript insère le code CSS dans l'en-tête de la page, on ne peut pas changer ce CSS par une feuille de style externe. C'est la raison pour laquelle la configuration TypeScript au sein du fichier `static/setup.txt` est enregistrée dans l'extension. Le fichier est intégré dans `ext_tables.php`, le rendant disponible en tant que gabarit statique.

```
// add TS/CSS to static templates
t3lib_extMgm::addStaticFile($_EXTKEY,'static/','dataMiner: CSS');
```

Le code CSS est inséré uniquement si le gabarit statique est inséré dans un enregistrement de gabarit.

Le code HTML du frontend a déjà très belle allure. La méthode `_load_javascript()`, qui intègre un fichier JavaScript à l'aide de la balise `<script>`, fait partie de la classe `dataMiner`. Cette méthode est adaptée, d'une part pour avoir des chemins de fichiers corrects, et d'autre part, pour que la balise apparaisse dans l'en-tête HTML.

```
_load_javascript() {
    print "<script language=\"javascript\" src=\"lib/dM.js\"></script>\n";

_load_javascript() {
    $GLOBALS['TSFE']->additionalHeaderData[$this->pObj->prefixId] =
        '<script type="text/javascript" src="'.
        $GLOBALS['TSFE']->absRefPrefix.t3lib_extMgm::siteRelPath('dataminer').
        'dataMiner/lib/dM.js"></script>';
}

```

Pour que cela fonctionne, l'objet `plugin` dans `tx_dataminer_pi1` doit être mis à la disposition de l'objet `dataMiner`.

Pour rendre les liens compatibles avec le plugin, un peu de travail manuel est malheureusement nécessaire. De simples remplacements de texte ne suffisent pas ici. Le paramètre `__b` est simplement omis, parce que `b` active le mode « affichage liste », qui est de toute façon le mode par défaut. À d'autres endroits, le tiret de soulignement est remplacé par `mode`, ce qui est plus explicite.

De plus, pour des raisons d'uniformisation, `page` est renommée `pointer` puisque, dans la classe `t3lib_pibase`, cette fonction est remplie par le paramètre `pointer`.

```
$string.= "<a href=\"".$_SERVER['PHP_SELF'].
    "?_b&page={$this->num_pages}\">".
    "<img src=\"img/last.gif\" border=\"0\" alt=\"Last\"></a>";

$icon = '';
$string.= $this->pObj->pi_linkTP_keepPIvars($icon,
    array('pointer'=>$this->num_pages), 0);
```

L'URL de destination doit bien entendu aussi être adaptée.

```
print '<form action="'.$_SERVER['PHP_SELF'] ...

print '<form action="'.$this->pObj->pi_linkTP_keepPIvars_url() ...
```

À ce stade, les liens sont corrects, mais les paramètres transférés ne sont pas reconnus. La solution à apporter est simple puisque les paramètres sont évalués dans dataMiner par la méthode `_get_value()`. Si vous utilisez `$this->pObj->piVars[]` ici, la plupart des paramètres seront reconnus correctement. Quelques ajustements pour les données POST doivent être faits dans la méthode `_var_setup()`.

Puisque `t3lib_pibase` prend complètement en charge le traitement des paramètres, on peut supprimer le code restant relatif aux sessions PHP.

Bien que la connexion à la base de données soit effective, il serait préférable de se passer de la couche Metabase. Puisque dataMiner encapsule l'accès à la base de données dans une classe distincte, nous devons aussi ici recourir aux fonctions de MySQL de la couche de base de données de TYPO3. Ces fonctions ne garantissent pas d'abstraction de la base de données, comme les fonctions `exec_*` de `t3lib_DB`, mais sont mises en œuvre dans cette application sans trop d'efforts.

```
$result = $this->query($query);
$row = $this->fetch_array($result);

$result = $GLOBALS['TYPO3_DB']->sql_query($query);
$row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($result);
```

dataMiner comporte de nombreuses options de configuration quant au choix des champs à éditer ou à afficher, ou quant à la dénomination de ces champs. Vous vous apercevrez très vite qu'il y a un parallélisme avec les définitions TCA. Il semble opportun de configurer dataMiner automatiquement à partir du TCA. Pour ce faire, l'extension **System language labels** (lang) est installée, puisqu'elle n'est pas disponible par défaut dans le frontend. Il faut déterminer les noms corrects des champs.

```
require_once(t3lib_extMgm::extPath('lang').'lang.php');
...
$LANG = t3lib_div::makeInstance('language');
...
$dm->fields = array_keys($TCA[$dm->table]['columns']);
$dm->title = $LANG->sL($TCA[$dm->table]['ctrl']['title']);
$dm->orderBy = $TCA[$dm->table]['ctrl']['sorting'];
```

```
foreach ($TCA[$dm->table]['columns'] as $column => $def) {
    $dm->humanize($column,$LANG->sL($def['label']));
    if($def['config']['default']) {
        $dm->defvalue($column,$def['config']['default']);
    }
}
...
```

Que reste-t-il à faire ? La configuration via TypoScript est souhaitable, ainsi que la possibilité de sélectionner les tables à afficher dans l'élément de contenu plugin.

Nos efforts pour porter le script en valent-ils la peine ? La réponse à cette question est difficile. Il aurait été possible d'intégrer le script plus facilement—mais aussi de façon moins correcte. Même une réécriture complète ou une solution mixte est envisageable. Avec dataMiner, vous disposez d'une application qui fonctionne déjà, vous permettant ainsi de procéder à des tests durant le processus de portage, ce qui constitue un grand avantage.

Au cours d'une dernière étape, vous pourriez supprimer le code redondant pour arriver à une fonctionnalité de type TCA, ce qui ferait de cette extension une application purement TYPO3.

7.7 Programmation du backend : principes

Par programmation backend, on entend la programmation de composants qui ajoutent des fonctionnalités dans le backend. Il s'agit en général de modules, de sous-modules ou d'entrées dans le menu contextuel.

Comme l'URL vous l'indique lorsque vous vous identifiez, le backend se trouve dans le répertoire `typo3/`. À propos, il est tout à fait possible d'effacer le backend d'une installation TYPO3 pour ne garder que le frontend.

Les modules sont normalement situés au sein de leurs propres extensions ; pour les modules standards, par contre, ce n'est pas toujours le cas pour des raisons historiques. Néanmoins, ces derniers sont correctement intégrés dans le système, et quelques-uns d'entre eux pourraient certainement se retrouver dans des extensions. D'autres modules se trouvent dans le répertoire `typo3/mod/` et d'autres encore directement dans `typo3/`. Le module **Web** → **Liste**, par exemple, a son propre fichier `conf.php` dans le répertoire `typo3/mod/web/list/` ; mais le script du module est en fait situé dans `typo3/db_list.php`. Il est correctement configuré via le fichier `conf.php`, et donc disponible dans le système. Ces exceptions mises à part, tous les nouveaux modules sont développés en tant qu'extensions.

7.7.1 Structure d'un module

Par opposition au frontend où toutes les requêtes transitent via un seul fichier — à savoir `index.php`, un lien symbolique vers `tslib/index_ts.php` —, dans le backend, chaque module dispose de son propre fichier PHP fournissant l'ensemble des fonctionnalités. Dans le cas du module **Web** → **Liste**, le fichier est `db_list.php`. Le nom de fichier du module dans les extensions est invariablement `index.php` puisque chaque module est inclus dans son propre répertoire.

En général, un module est composé des fichiers suivants :

```

conf.php
index.php
locallang.php
locallang_mod.php
moduleicon.gif

```

conf.php

Le module est intégré dans le système via ce fichier. Il est créé par le Kickstarter et ne doit normalement pas être modifié.

```

define('TYPO3_MOD_PATH', '../typo3conf/ext/user_recentchanges/mod1/');
$BACK_PATH='../typo3/';
$MCONF['name'] = 'tools_urecentchangesM1';
$MCONF['access'] = 'admin';
$MCONF['script'] = 'index.php';
$MLANG['default']['tabs_images']['tab'] = 'moduleicon.gif';
$MLANG['default']['ll_ref'] =
    'LLL:EXT:user_recentchanges/mod1/locallang_mod.php';

```

En premier lieu, la constante `TYPO3_MOD_PATH` est définie. Elle contient le chemin vers le module depuis le répertoire `typo3/`.

La variable `$BACK_PATH` contient le chemin relatif vers le répertoire `typo3/`. Grâce à cette variable, on peut rendre les modules indépendants de leur position dans la hiérarchie des répertoires. Le gestionnaire d'extensions adapte ces chemins dans les deux premières lignes durant l'installation. C'est la raison pour laquelle les extensions contenant des modules ne peuvent pas être déplacées manuellement vers le répertoire d'une autre extension.

La clé `name` du tableau `$MCONF` définit un nom unique pour le module. La clé `access` spécifie les permissions d'accès au module. Alors que la valeur `admin` limite l'accès aux seuls administrateurs, le module est rendu disponible pour les utilisateurs normaux avec `user,group`. Mais l'accès en lui-même est en définitive contrôlé par la configuration des enregistrements pour les utilisateurs et les groupes d'utilisateurs.

L'icône qui apparaît dans la barre des modules est insérée dans le tableau `$MLANG`. La référence à un fichier externe pour les langues est aussi spécifiée.

locallang_mod.php

Ce fichier est aussi créé par le Kickstarter et contient le titre et la description du module dans différentes langues. Les clés suivantes sont utilisées à cet effet :

mlang_tabs_tab

Un titre court pour la navigation

mlang_labels_tablabel

Un titre un peu plus long comme description du titre court ou comme en-tête dans **Aide** → **sur les modules**

mlang_labels_tabdescr

Description du module en plusieurs phrases ; est aussi repris dans **Aide** → **sur les modules**

index.php

Le script du module lui-même

locallang.php

Le fichier de langue associé au script du module.

Le seul fichier absolument indispensable au module est **conf.php**. Tous les autres fichiers peuvent porter d'autres noms et se trouver à d'autres endroits, pour autant qu'ils soient correctement définis dans **conf.php**. Mais puisque le Kickstarter met en place ces fichiers sous cette forme, et que des déviations par rapport à ce schéma n'apportent pas plus de clarté, vous devriez essayer d'adopter cette structure autant que possible.

```
t3lib_extMgm::addModule('web',  
                        'ushopinfoM1',  
                        '',  
                        t3lib_extMgm::extPath($_EXTKEY) . 'mod1/');
```

Dans l'exemple ci-dessus, le module **ushopinfoM1** est inséré dans le module principal **web**.

7.7.2 Module: framework

Les fichiers et les classes suivants sont insérés par les modules :

init.php

Ce fichier doit être intégré par chaque module. Un certain nombre de bibliothèques sont intégrées, l'utilisateur est identifié, la configuration du système est chargée et les chemins sont initialisés.

Les constantes suivantes sont définies après l'appel de **init.php** :

PATH_thisScript

Chemin absolu vers le script du module

PATH_typo3

Chemin absolu vers le backend TYPO3

PATH_typo3_mod

Chemin relatif vers le module depuis **typo3/**

PATH_site

Chemin absolu vers le site Web

PATH_t3lib

Chemin absolu vers **t3lib/**

Les objets globaux sont disponibles après que **init.php** a été appelé :

\$BE_USER

Utilisateur backend

\$LANG

Gestion des langues

\$TYPO3_DB

Accès à la base de données

t3lib_SCbase

C'est la classe de base pour les scripts des modules. En particulier, elle fournit un framework complet pour le support des fonctions de sous-modules. De plus, cette classe gère automatiquement la configuration d'un module, des menus, et le passage des paramètres. Son fonctionnement est décrit plus en détail grâce aux exemples suivants.

Les variables suivantes dans les scripts des modules sont :

\$this->id

Contient l'ID de la page courante tant que le module est situé dans le module principal **Web** ; les modules de la zone **Fichiers** trouvent ici le chemin courant.

\$this->CMD

Est fixée par `t3lib_div::_GP('CMD')` et est disponible pour votre propre utilisation.

\$this->perms_clause

Contient une clause SQL WHERE sur la table **pages**, qui filtre les pages pour lesquelles l'utilisateur courant n'a pas de permission.

\$this->MOD_MENU

Tableau des menus ; son utilisation sera illustrée par les exemples suivants.

\$this->MOD_SETTINGS

Le tableau pour la configuration des menus.

\$this->modTSconfig

Les modules TSConfig, basés sur le TSConfig des pages et le TSConfig Utilisateur.

template, smallDoc, mediumDoc, bigDoc

Le fichier `typo3/template.php` contient plusieurs classes pour la restitution des modules. Les différentes classes sont cependant toutes des variantes de **template**, définissant des largeurs de sorties, puisque, à cause du cadre de navigation, les modules dans **Web** doivent être de largeur plus petite que les modules dans **Outils**. Par convention, `$this->doc` dans les modules contient une instance de **template**, ou d'une de ses variantes. Le Kickstarter utilise déjà `$this->doc` dans le code qu'il génère :

```
// Ouput page header
$this->content.=$this->doc->startPage($LANG->getLL('title'));
$this->content.=$this->doc->header($LANG->getLL('title'));
$this->content.=$this->doc->spacer(5);
```

t3lib_BEfunc

Cette classe contient des fonctions utiles pour la programmation backend, c'est-à-dire pour l'accès à la base de données, le cache, TSConfig, l'arborescence et l'affichage des modules. La fonction de menu est particulièrement utilisée. Cette classe, tout comme les deux suivantes, ne sont pas instanciées, mais leurs fonctions sont appelées directement :

```
t3lib_BEfunc::getFuncMenu($this->id,
    'SET[function]',
    $this->MOD_SETTINGS['function'],
    $this->MOD_MENU['function']);
```

t3lib_div

La série de fonctions `t3lib_div` est bien sûr aussi à votre disposition pour le développement dans le backend.

t3lib_iconworks

Ensemble de fonctions pour la création et la fourniture d'icônes ; par exemple :

```
t3lib_iconWorks::getIconImage('tt_content_search', array(), $BACK_PATH)
```

```
t3lib_iconworks::getIconImage($table, $row, $BACK_PATH,  
    'class="c-recicon" title="'. $iconAltText. '"')
```

La fonction `t3lib_iconWorks::skinImg()` détermine le nom d'un fichier en fonction d'une icône et d'un habillage donnés. Vous devriez aussi utiliser cette fonction pour vos propres icônes, pour pouvoir habiller librement votre application.

Pour les détails sur les classes de l'API, référez-vous par exemple à la documentation sur l'API disponible dans l'extension `ExtDevEval`.

7.7.3 Modules : script

La structure de base d'un script de module ressemble habituellement à ceci :

```
unset($MCONF);  
require ('conf.php');  
require ($BACK_PATH.'init.php');  
require ($BACK_PATH.'template.php');  
  
$LANG->includeLLFile('EXT:user_example/mod1/locallang.php');  
  
require_once (PATH_t3lib.'class.t3lib_scbase.php');  
  
$BE_USER->modAccess($MCONF,1);  
  
class user_example_module1 extends t3lib_SCbase {  
  
    function menuConfig() {  
    }  
  
    function main() {  
    }  
  
    function printContent() {  
    }  
}  
  
$SOBE = t3lib_div::makeInstance('user_example_module1');  
$SOBE->init();  
$SOBE->main();  
$SOBE->printContent();
```

On intègre d'abord le fichier `conf.php`. Comme nous l'avons déjà mentionné, `conf.php` ne contient pas seulement de l'information importante pour le backend, mais aussi la variable

`$BACK_PATH`, qui spécifie le chemin relatif au répertoire `typo3/`, auquel on fait fréquemment référence dans le script.

Ensuite, le fichier `init.php` du répertoire `typo3/` est intégré. Ce fichier initialise l'environnement du module et est absolument indispensable. Le prochain fichier à être intégré, `template.php`, contient une bibliothèque pour l'affichage du module.

À présent, l'objet pour la gestion des langues est disponible dans `$LANG`. La définition des langues pour le module est chargée et initialisée.

La classe de base du module `t3lib_SCbase` est alors intégrée. Le module est une classe qui étend `t3lib_SCbase`. La méthode `menuConfig()` définit les menus et les options pour le module. Elle est appelée par la classe de base.

Pour finir, à la fin du script, une instance de la classe de module est créée et appelée. C'est aussi la raison pour laquelle on les appelle « SC » (`t3lib_SCbase`), c'est-à-dire les classes de script (*Script Classes*) : elles contiennent la classe du module, mais cette classe est aussi appelée. Le nom `$SOBE` est obligatoire car des bibliothèques externes peuvent y faire référence.

7.7.4 Module principal

Il est aussi possible de créer, par exemple via le Kickstarter, un nouveau module principal tel que **Web**, **Utilisateur** ou **Outils**. C'est supporté par le Kickstarter. La création d'un module principal avec un cadre de navigation (tel que **Web** ou **Fichier**) n'est pas encore supportée par le Kickstarter, mais est possible. Voici un exemple du module principal **Media** et du module **Media** → **Liste**.

mod1/conf.php

```
define('TYPO3_MOD_PATH', '../typo3conf/ext/dam/mod1/');
$BACK_PATH='../typo3/';

$MCONF['name']='txdamM1';
$MCONF['access']='user,group';
$MCONF['navFrameScript']='alt_dam_navframe.php';
$MCONF['defaultMod']='list';
```

La clé `navFrameScript` définit le script à appeler dans le cadre de navigation. Le cadre lui-même est créé par le backend.

mod1/list/conf.php

```
define('TYPO3_MOD_PATH', '../typo3conf/ext/dam/mod1/list/');
$BACK_PATH='../typo3/';

$MCONF['name']='txdamM1_list';
$MCONF['access']='group,user';
$MCONF['script']='index.php';
```

ext_tables.php

```
t3lib_extMgm::addModule('txdamM1', '', '',
    t3lib_extMgm::extPath('dam').'mod1/');
t3lib_extMgm::addModule('txdamM1', 'list', '',
    t3lib_extMgm::extPath('dam').'mod1/list/');
```


C'est ici qu'on déclare les modules au système : d'abord le module principal `txdamM1` et ensuite le module list en tant que sous-module.

7.7.5 Fonctions de sous-modules

Plusieurs modules tels que **Web** → **Fonctions** ou **Utilisateur** → **Centre de tâches** offrent la possibilité d'incorporer des fonctions. Ces fonctions de sous-modules peuvent être des extensions séparées. Après l'installation, elles s'affichent dans le menu du module correspondant. Certaines fonctions de sous-module sont discutées plus en détail dans l'exemple **Web** → **Fonctions** → **Assistants**.

7.8 Programmation backend : exemple

Les exemples suivants donnent un aperçu des possibilités existantes permettant d'ajouter de nouvelles fonctionnalités au backend.

Pour vos propres projets, le Kickstarter vous fournit déjà un cadre adéquat. Si certaines fonctions manquent, consultez d'autres modules pour voir si vous pouvez y retrouver des fonctions similaires. Si vous cliquez sur le bouton droit de votre souris dans le cadre du module, la plupart des navigateurs vous donneront des informations sur l'URL sélectionnée, ce qui doit vous indiquer le répertoire dans lequel se trouve le script du module. Vous pouvez dès lors étudier le code source de ce module.

La version 3.8 de TYPO3 a été utilisée pour les exemples backend, comme ce fut le cas pour les exemples frontend.

7.8.1 Outils → Dernières modifications

Le premier exemple de programmation backend est un module qui affiche les dernières modifications apportées aux pages. Seuls les administrateurs auront accès au module dans **Outils**. Ils auront quelques options d'affichage sous la forme d'un menu de sélection et d'une case à cocher. De plus, les enregistrements et les pays sont affichés avec des liens qui vous redirigent respectivement vers le module des pages ou vers le formulaire de modification de l'enregistrement.

Figure 7.39:
Le module Dernières
modifications



On crée d'abord une extension en tant que module dans le Kickstarter. Le module est sauvé sous **Tools** et l'option **Admin-only access !** est activée.

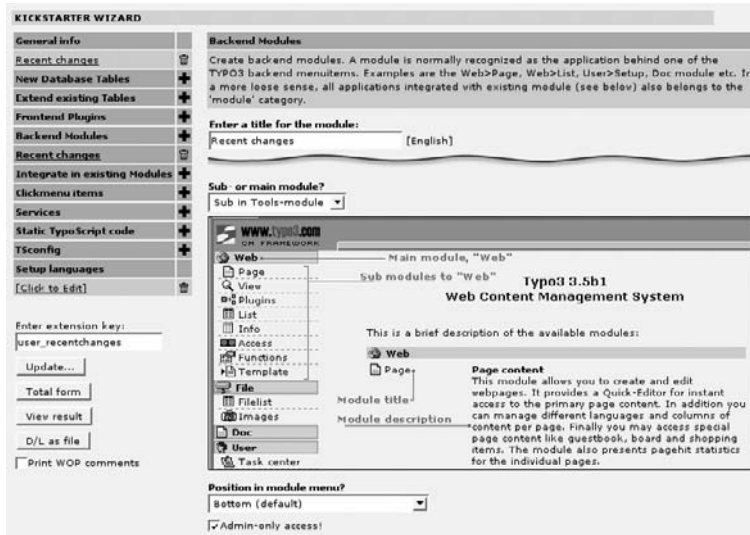


Figure 7.40:
Création du module
avec l'extension
Kickstarter

Le module est rapidement configuré puisque seules quelques options doivent être activées. Le Kickstarter crée les fichiers suivants :

```
ext_emconf.php
ext_icon.gif
ext_tables.php
mod1/
  clear.gif
  conf.php
  index.php
  locallang.php
  locallang_mod.php
  moduleicon.gif
```

Le module est déclaré au système dans le fichier `ext_tables.php`.

```
t3lib_extMgm::addModule('tools', 'urecentchangesM1', '',
    t3lib_extMgm::extPath($_EXTKEY) . 'mod1/');
```

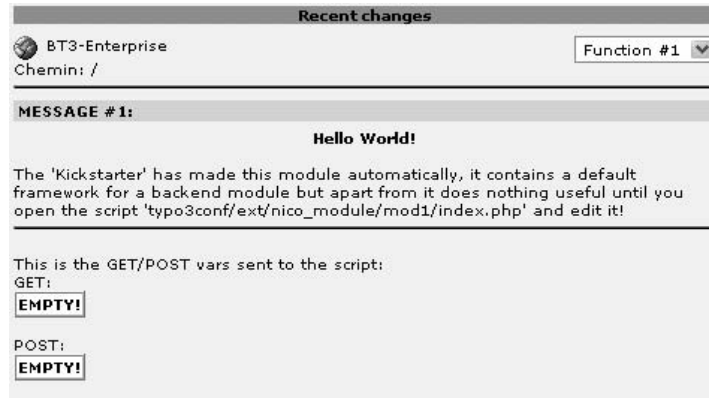
Le fichier `mod1/index.php` contient le module. Si le module vient d'être créé et installé, il produit déjà l'affichage par défaut repris à la figure 7.41.

Dans le fichier `index.php` se trouve déjà la structure de base du module. Un menu et quelques exemples d'affichage ont déjà été générés.

Ci-dessous, vous retrouvez le code du module terminé. Il est recommandé de créer le module via le Kickstarter, pour que vous puissiez voir les différences. Le concept du module consiste à sélectionner l'information de la table `sys_log` qui fournit les détails sur les pages et les enregistrements récemment modifiés, et qui les affichent dans une table. Vous y trouverez également

d'autres informations telles que les icônes, les titres, le rootline, et l'aide contextuelle avec les ID.

Figure 7.41:
Affichage du module
après sa création par
le Kickstarter



En premier lieu, le module est initialisé par `index.php`. Par opposition au Kickstarter, aucun changement n'est ici nécessaire.

```
// DEFAULT initialization of a module [BEGIN]
unset($MCONF);
require ('conf.php');
require ($BACK_PATH.'init.php');
require ($BACK_PATH.'template.php');

// Include locallang file for module
$LANG->includeLLFile('EXT:user_recentchanges/mod1/locallang.php');

// Include module base class
require_once (PATH_t3lib.'class.t3lib_sbase.php');

// This checks permissions
// and exits if the users has no permission for entry.
$BE_USER->modAccess($MCONF,1);
// DEFAULT initialization of a module [END]

class user_recentchanges_module1 extends t3lib_SCbase {
```

La première méthode de la classe du module surcharge `menuConfig()` de `t3lib_SCbase`. Les définitions de tous les menus et des options du module sont situées dans `$this->MOD_MENU[]`. Typiquement, la clé `function` contient la définition du menu principal, qui est généralement restitué sous forme d'un champ de menu de sélection. Le Kickstarter l'a déjà créé à titre d'exemple. Les options pour afficher uniquement les nouveaux contenus, les contenus modifiés ou les deux à la fois doivent être inclus dans le menu principal du module. Le tableau de menu est rempli par les clés correspondantes.

```
/**
 * Adds items to the ->MOD_MENU array.
 * Used for the function menu selector.
```

```

*
* @return void
*/
function menuConfig() {
    global $LANG;

    $this->MOD_MENU = Array (
        'function' => Array (
            'newAndUpdated' => $LANG->getLL('newAndUpdated'),
            'updated' => $LANG->getLL('updated'),
            'new' => $LANG->getLL('new'),
        ),
        'showRootline' => true,
    );
    parent::menuConfig();
}

```

Les textes des éléments de menu sont traduits vers les langues de l'utilisateur via `$LANG->getLL()`, si la langue existe. Les définitions dans les différentes langues se trouvent dans `mod1/locallang.php` :

```

$LOCAL_LANG = Array (
    'default' => Array (
        'title' => 'Recent changes',
    ...
    'fr' => Array (
        'title' => 'Dernières modifications',
        'showRootline' => 'Afficher rootline',
        'new' => 'Contenu créé',
        'updated' => 'Contenu modifié',
        'newAndUpdated' => 'Contenu créé et modifié',
        'new_header' => 'Dernier contenu créé:',
        'updated_header' => 'Dernier contenu modifié:',
        'newAndUpdated_header' => 'Dernier contenu créé et modifié:',
        'admins_only' => 'Module réservé aux admins!',
    ),
)

```

Pour des raisons de simplicité et de clarté, et même si ce n'est pas nécessaire, nous avons utilisé la même clé pour le menu et pour la définition des langues.

Retour au code du module. Outre `function`, `$this->MOD_MENU` contient aussi l'entrée `showRootline`, une valeur booléenne qui est utilisée comme case à cocher. Pour finir, `parent::menuConfig()` est appelé pour terminer la configuration du menu.

La méthode suivante est `main()`, la méthode principale du script du module. Le nom répond à une convention, même si ce n'est pas obligatoire. Quelques initialisations du module sont effectuées ici. L'affichage du module est généré, par souci de clarté, par `moduleContent()`, qui ne se situe pas à l'intérieur de cette méthode.

```

/**
 * Main function of the module. Write the content to $this->content
 *
 * @return void

```

```
*/
function main() {
    global $BE_USER,$LANG,$BACK_PATH;
```

Tout d'abord, on vérifie si l'utilisateur a bien les droits d'administrateur.

```
if ($BE_USER->user['admin']) {
```

Dans le code créé par le Kickstarter, cette ligne est un peu différente. De plus, on y initialise `$this->pageinfo`, qui se réfère à la page courante sélectionnée. Le code Kickstarter est mis en place pour être utilisé dans le module Web. Mais, puisque le module fonctionne dans le module principal **Outils**, dans lequel le concept de page sélectionnée n'a pas de sens, cette partie du code a été supprimée.

Ensuite, un objet **template** ou plus précisément **bigDoc** est créé dans `$this->doc` (cf. section 7.7.2).

```
// Init the module doc
$this->doc = t3lib_div::makeInstance('bigDoc');
$this->doc->backPath = $BACK_PATH;
$this->doc->form = '<form action="" method="POST">';
```

Le code JavaScript qui suit a été généré par le Kickstarter et est nécessaire à la fonction des menus :

```
// JavaScript, used for menus
$this->doc->JScode = $this->doc->wrapScriptTags('
    function jumpToUrl(URL) {
        document.location = URL;
    }
');
```

La page HTML à afficher est initialisée via l'objet **template**, et le titre du module est affiché. Le tout est enregistré dans la variable `$this->content`.

```
// Ouput page header
$this->content.= $this->doc->startPage($LANG->getLL('title'));
$this->content.= $this->doc->header($LANG->getLL('title'));
$this->content.= $this->doc->spacer(5);
```

Le menu est créé. On utilise à cette fin les fonctions de **t3lib_BEfunc**. On passe `$this->id` en argument à ces fonctions. Pour les modules de la zone **Web**, cette variable contient l'ID de la page sélectionnée. Cette dernière est envoyée au script courant par la fonction de menu via le paramètre `id`, qui est reconnu par **t3lib_SCbase**. Ceci n'a bien sûr pas de sens dans un module de la zone **Outils**, puisque l'arborescence n'est pas disponible dans ce cas, et que le module est supposé afficher uniquement les pages qui ont changé. Nous laissons néanmoins ce code, puisqu'il ne dérange en rien, et qu'il pourrait être utile si le module devait être placé plus tard dans la zone **Web**.

```
// Output menu
$menu = array();
```

```
$menu[] = t3lib_BEfunc::getFuncMenu($this->id,
                                     'SET[function]',
                                     $this->MOD_SETTINGS['function'],
                                     $this->MOD_MENU['function']);
```

SET[function] définit le paramètre qui permet de passer la valeur du menu au script du module. \$this->MOD_SETTINGS['function'] contient la valeur courante et \$this->MOD_MENU['function'] est la définition du menu sous forme d'un tableau qui contient les valeurs et les textes que l'utilisateur visualisera. En voici le comportement : un menu de sélection est créé, qui, grâce à la fonction JavaScript `jumpToUrl()` ajoutée ci-dessus, envoie la valeur du menu sélectionné par l'utilisateur au script du module avec le paramètre SET[function]. Le paramètre SET est évalué par la classe de base du module `t3lib_SCbase`, comparé avec \$this->MOD_MENU[] et sauvegardé dans \$this->MOD_SETTINGS[]. Pour finir, cette configuration du module est enregistrée dans la base de données, pour qu'à la prochaine visite, l'utilisateur retrouve le module dans l'état où il l'a laissé.

La case à cocher se comporte de la même manière.

```
$menu[] = t3lib_BEfunc::getFuncCheck($this->id,
                                     'SET[showRootline]',
                                     $this->MOD_SETTINGS['showRootline']).' '.
                                     $LANG->getLL('showRootline');
$this->content.= $this->doc->section(' ', implode('<br />', $menu));
$this->content.= $this->doc->divider(5);
```

Le contenu du module est créé en appelant la méthode `moduleContent()`.

```
// Render content:
$this->moduleContent();
```

Pour finir, l'icône de raccourci est affichée.

```
// ShortCut
if ($BE_USER->mayMakeShortcut()) {
    $this->content.= $this->doc->spacer(20).
        $this->doc->section(' ',
            $this->doc->makeShortcutIcon('id',
                implode(' ', array_keys($this->MOD_MENU)),
                $this->MCONF['name'])
        );
}

$this->content.= $this->doc->spacer(10);
```

Si l'utilisateur n'est pas administrateur, il ou elle reçoit l'avertissement suivant :

```
} else {
    // If no access: output message

    $this->doc = t3lib_div::makeInstance('bigDoc');
    $this->doc->backPath = $BACK_PATH;
```

```

        $this->content.= $this->doc->startPage($LANG->getLL('title'));
        $this->content.= $this->doc->header($LANG->getLL('title'));
        $this->content.= $this->doc->spacer(5);
        $this->content.= $this->doc->section('',
                                $LANG->getLL('admins_only'));
        $this->content.= $this->doc->spacer(10);
    }
}

```

La méthode suivante, `printContent()`, a été créée par le Kickstarter. Lorsqu'elle est appelée, la page HTML générée est finalement affichée, ce qui a lieu à la fin du fichier.

```

/**
 * Prints out the module's HTML code
 *
 * @return void
 */
function printContent() {
    $this->content.= $this->doc->endPage();
    echo $this->content;
}

```

La méthode `moduleContent()` est aussi insérée par le Kickstarter. Dans de nombreux modules, elle contient une construction de type `switch()` qui permet d'appeler différentes fonctions du module au moyen de `$this->MOD_SETTINGS['function']` ou d'autres options. Dans ce cas, la valeur de `$this->MOD_SETTINGS['function']` est d'abord déterminée par la méthode `$this->getRecentChangesTable()`. La valeur `_header` est aussi définie, de sorte que le texte de l'entête est déterminé via sa valeur dans `locallang.php`.

```

/**
 * Generates the module content
 *
 * @return void
 */
function moduleContent() {
    global $LANG;

    $mode = $this->MOD_SETTINGS['function'];
    $content = $this->getRecentChangesTable($mode);
    $this->content.= $this->doc->section($LANG->getLL($mode.'_header'),
                                    $content, 0, 1);
}

/**
 * Render the table with recently changed records
 *
 * @param string Mode from $this->MOD_SETTINGS['function']
 * @return string Rendered Table
 */
function getRecentChangesTable($mode) {
    global $BACK_PATH, $BE_USER, $LANG, $TCA;

```

En utilisant cette méthode, le tableau résultat, reprenant les dernières pages éditées, est créé. Le paramètre `$mode` passé en argument détermine la valeur de la variable `$action`, qui filtre les entrées de la table `sys_log` grâce à la requête suivante :

```
// Set sys_log actions depending on selected mode
if ($mode=='new') {
    $action = '1';
} elseif ($mode=='updated') {
    $action = '2';
} else {
    $action = '1,2';
}

// Query sys_log for non-deleted pages only
$res = $GLOBALS['TYPO3_DB']->exec_SELECTquery(
    'sys_log.*',
    'sys_log,pages',
    'pages.uid=sys_log.event_pid'.
    ' AND sys_log.event_pid>0'.
    ' AND sys_log.type=1'.
    ' AND sys_log.action IN (\'$action.\')'.
    ' AND sys_log.error=0'.
    t3lib_BEfunc::deleteClause('pages'),
    '',
    'tstamp DESC',
    40);
```

Pour l'affichage du tableau, on utilise la méthode `$this->doc->table()`, qui restitue un tableau en se basant sur la définition suivante. Le tableau `$table` est utilisé pour contenir les données de la table.

```
// init table layout
$tableLayout = array (
    'table' => array ('<table border="0" cellpadding="1" cellspacing="1" class="typo3-recent-edited">',
    '</table>'),
    'defRow' => array (
        'tr' => array ('<tr class="bgColor4">', '</tr>'),
        'defCol' => Array ('<td valign="top">', '</td>')
    )
);

$table = array();
$tr = 0;
```

Dans la boucle suivante, les entrées de la table `sys_log` sont traitées pour être restituées dans le tableau.

```
while($logRow = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res)) {

    $page_id = $logRow['event_pid'];
    $pageRow = t3lib_BEfunc::getRecord('pages', $page_id);
```


Tout d'abord, on retrouve l'enregistrement de la page à partir de l'ID dans la table `sys_log`. Pour la première colonne du tableau à afficher, un lien est généré avec l'icône par la méthode `getItemFromRecord()` pour l'édition de la page (`$contentPageLink`). Pour la dernière colonne, l'heure de la dernière modification (`$contentAge`) est générée.

```
if (is_array($pageRow)) {

    // Create output item for pages record
    $contentPageLink = $this->getItemFromRecord('pages', $pageRow);

    // Create output text describing the age
    $contentAge = t3lib_BEfunc::dateTimeAge($logRow['tstamp'], 1);
```

Ensuite, les détails concernant l'élément modifié de la page sont créés.

```
$contentElementLink = '';
$contentUser = '';

// Fetch record if table is not "pages"
if(!($logRow['tablename']=='pages')) {
    $elementRow = t3lib_BEfunc::getRecord(
                                                $logRow['tablename'],
                                                $logRow['recuid']);
```

L'enregistrement `sys_log` contient de l'information sur la table et l'uid de l'enregistrement modifié. On fait une recherche sur cet uid ; s'il existe, un lien pour l'édition du record est créé dans la seconde colonne (`$contentElementLink`).

```
// If record is deleted continue with next log entry
if (!is_array($elementRow)) {
    continue;
}

// Create output item for non pages record
if(!($logRow['tablename']=='pages')) {
    $contentElementLink = $this->getItemFromRecord(
                                                $logRow['tablename'],
                                                $elementRow);
}
}
```

La table contient aussi de l'information à propos de l'utilisateur qui est intervenu sur la page. On sélectionne d'abord l'enregistrement depuis la table `be_users` et ensuite, un lien pour son édition est créé (`$contentUser`).

```
// Create user item
$userRow = t3lib_BEfunc::getRecord('be_users',
                                    $logRow['userid']);

if(is_array($userRow)) {
    $contentUser = htmlspecialchars($userRow['username'].
                                    ' ('. $userRow['uid']. ')');
    $contentUser = $this->wrapEditLink($contentUser,
                                    'be_users',
                                    $userRow['uid']);
```

Si l'utilisateur est l'utilisateur courant, une couleur d'arrière-fond différente est utilisée dans le tableau `$tableLayout` pour la ligne en question, pour mettre en évidence le fait que cette page a été modifiée par l'utilisateur lui-même.

```
// Use different row color
// if record was edited by current user
if ($userRow['uid']==$BE_USER->user['uid']) {
    $tableLayout[$str]['tr'] = array(
        '<tr class="bgColor5">',
        '</tr>');
}
}
```

Pour finir, les données sont écrites dans le tableau `$table`. Après la boucle, c'est-à-dire après avoir traité toutes les entrées de `sys_log`, on restitue le tableau avec `$this->doc->table()`. Vous pourriez bien sûr simplement construire la table HTML par des chaînes de caractères – c'est juste une question de goût.

```
// Add row to table
$td=0;
$table[$str][$td++] = $contentPageLink;
$table[$str][$td++] = $contentElementLink;
$table[$str][$td++] = $contentUser;
$table[$str][$td++] = $contentAge;
$tr++;
}

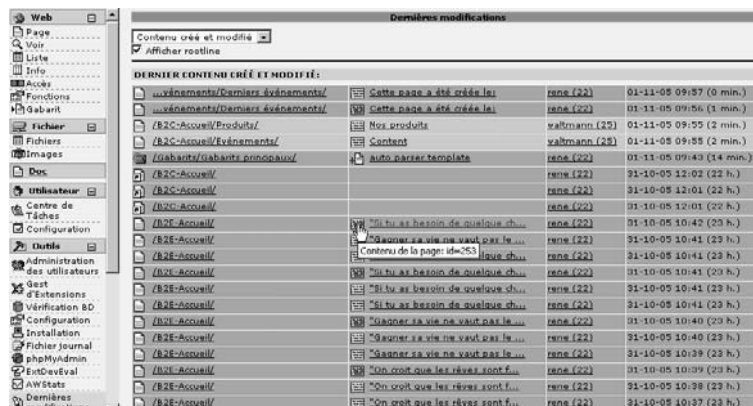
// Return rendered table
return $this->doc->table($table, $tableLayout);
}
```

La méthode suivante crée un élément d'enregistrement pour son affichage dans le tableau en fonction du nom de la table et du tableau contenant les données de l'enregistrement.

```
/**
 * Returns a linked icon with title from a record
 *
 * @param string Table name (tt_content,...)
 * @param array Record array
 * @return string Rendered icon
 */
function getItemFromRecord($table, $row) {
    global $BACK_PATH, $LANG, $TCA, $BE_USER;
```

Tout d'abord, le texte de l'attribut `title` (aide contextuelle) de l'icône est créé. L'aide contextuelle s'affiche si vous positionnez la souris sur l'icône.

Figure 7.42:
Le module **Dernières
modifications**
terminé

[illegible]

Pour montrer clairement de quel type d'enregistrement il s'agit, c'est-à-dire à quelle table il appartient, le texte est précédé du titre de la table, pour autant qu'il ne s'agisse pas de la table pages.

Le titre de l'enregistrement est alors déterminé. Les définitions TCA contiennent de l'information sur les champs de la base de données à afficher en tant que titre. Sur base de cette information, la fonction `t3lib_BEfunc::getRecordTitle()` détermine le titre. Aussi longtemps que l'option `showRootline` a été activée via la case à cocher, le chemin de rootline de la page est déduit et, si nécessaire, raccourci selon la longueur que l'utilisateur a spécifiée dans **Utilisateur** → **Configuration**.

```
// Create record title or rootline for pages
// if option is selected
if($table=='pages' AND $this->MOD_SETTINGS['showRootline']) {
    $elementTitle = t3lib_BEfunc::getRecordPath($row['uid'],
                                                '1=1',
                                                0);
    $elementTitle = t3lib_div::fixed_lgd_cs($elementTitle,
                                            -($BE_USER->uc['titleLen']));
} else {
    $elementTitle = t3lib_BEfunc::getRecordTitle($table, $row, 1);
}
```

Enfin, l'icône générée avec le titre est restituée avec un lien vers l'édition et renvoyée.

```
// Create icon for record
$elementIcon = t3lib_iconworks::getIconImage($stable, $row,
```

```

$BACK_PATH,
'class="c-recicon" title="'. $iconAltText.'";

    // Return item with edit link
    return $this->wrapEditLink($elementIcon.$elementTitle,
                              $table,
                              $row['uid']);
}

```

La dernière méthode du module crée un lien qui soit appelle le module des pages (pour les pages) via JavaScript, soit affiche directement un enregistrement dans un mode d'édition (pour les enregistrements des autres tables).

```

/**
 * Wraps an edit link around a string.
 * Creates a page module link for pages, edit link for other tables.
 *
 * @param string      The string to be wrapped
 * @param string      Table name (tt_content,...)
 * @param integer     uid of the record
 * @return string     Rendered link
 */
function wrapEditLink($str, $table, $id) {
    global $BACK_PATH;

    if($table=='pages') {
        $editOnClick = "top.fsMod.recentIds['web']=".$id.";";
        $editOnClick.= "top.goToModule('web_layout',1);";
    } else {
        $params = '&edit['.$table.']['.$id.']=edit';
        $editOnClick = t3lib_BEfunc::editOnClick($params, $BACK_PATH);
    }
    return '<a href="#" onclick="'.htmlspecialchars($editOnClick).'">'.
        $str.'</a>';
}

```

Comme vous le remarquez, le module des pages est appelé via une fonction JavaScript à partir du cadre principal. L'ID de la page a été fixé préalablement, et est passé au module lorsque ce dernier est appelé.

Le lien pour éditer un enregistrement est créé avec les paramètres correspondants par la fonction `t3lib_BEfunc::editOnClick()`.

Enfin, à la fin du fichier, on instancie et on appelle la classe du module.

```

// Make instance:
$SOBE = t3lib_div::makeInstance('user_recentchanges_module1');
$SOBE->init();

// Include files?
foreach($SOBE->include_once as $INC_FILE)    include_once($INC_FILE);

$SOBE->main();
$SOBE->printContent();

```

La structure d'un module permet d'écrire facilement des extensions. Il est simple d'ajouter des éléments de menu ou de nouvelles options. Par exemple, une vue détaillée pourrait être développée avec une méthode séparée et être appelée depuis `moduleContent()`.

7.8.2 Fonction de sous-module Web → Fonctions → Assistants

TYP03 propose la classe `t3lib_extobjbase` comme base des fonctions de sous-modules. La classe de base `t3lib_SCbase` contient une interface vers `t3lib_extobjbase`. Grâce à cette combinaison, il est relativement aisé de construire des modules qui comprennent des fonctions de sous-modules et les sous-modules eux-mêmes.

Dans l'exemple ci-dessous, une fonction de sous-module va être incorporée dans **Web → Fonctions → Assistants**. La fonction va marquer de manière récursive les pages pour activer ou désactiver la recherche. Cette option, appelée **Sans recherche**, est normalement disponible dans l'en-tête de la page.

Figure 7.43:
Nouvel assistant en
tant que sous-module
de Web → Fonctions



L'affichage de la fonction de sous-module démarre uniquement avec **Aktivieren la recherche sur les pages**, puisque cette fonction est incorporée dans **Fonctions → Assistants**. Chaque module qui permet les fonctions de module fournit, si nécessaire, un cadre ou des fonctions de base, qui peuvent prendre plusieurs formes, selon la fonctionnalité du module. Dans ce cas, le module affiche un menu de sélection de fonctions de sous-module, ainsi qu'un en-tête avec la page courante que l'utilisateur a sélectionnée via l'arborescence du cadre de navigation dans le module **Web**. Cette page est la page de départ des assistants.

La nouvelle fonction de sous-module utilise la page courante comme point de départ à partir duquel on parcourt l'arborescence. Le nombre de niveaux de l'arborescence qui seront parcourus peut être spécifié par l'utilisateur grâce à un champ de sélection. Les pages sont aussi affichées sous forme d'arborescence et leur statut **Sans recherche** est indiqué pour chacune d'elles. Les pages pour lesquelles l'utilisateur n'a pas le droit de changer le statut sont mises en surbrillance. Pour finir, il ou elle peut modifier le statut de toutes les pages affichées avec seulement deux boutons.

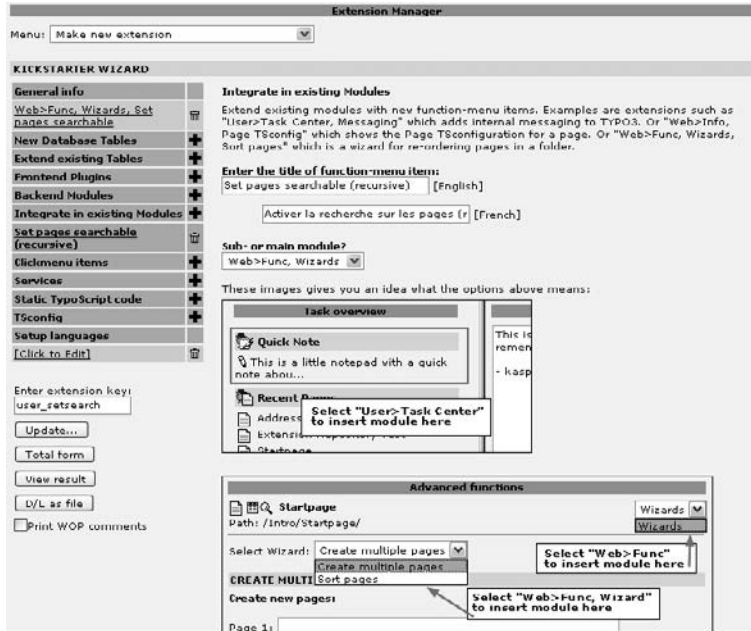


Figure 7.44:
Création d'une
fonction de
sous-module Web →
Fonctions →
Assistant par le
Kickstarter

Puisque le Kickstarter supporte directement les fonctions de sous-module dans **Web → Fonctions → Assistants**, un cadre de base est rapidement mis en place.

Exceptés le composant pour le français, la langue supplémentaire, et le composant **General info**, l'extension contient uniquement une fonction de sous-module qui est créée par l'option **Integrate in existing Modules**.

Le Kickstarter génère les fichiers suivants :

```
ext_emconf.php
ext_icon.gif
ext_tables.php
locallang_db.php
modfunc1/
    class.user_setsearch_modfunc1.php
    locallang.php
```

Les fichiers `locallang` contiennent les différentes langues, comme d'habitude. Dans le fichier `ext_tables.php`, la fonction de sous-module est déclarée :

```

t3lib_extMgm::insertModuleFunction(
    'web_func',
    'user_setsearch_modfunc1',
    t3lib_extMgm::extPath($_EXTKEY).
    'modfunc1/class.user_setsearch_modfunc1.php',
    'LLL:EXT:user_setsearch/locallang_db.php:moduleFunction.user_setsearch_modfunc1',
    'wiz'
);

```

Le Kickstarter a créé dans `class.user_setsearch_modfunc1.php` un bout de code exemple dans lequel une option de menu a été introduite et affichée en tant que case à cocher.

```

require_once(PATH_t3lib.'class.t3lib_extobjbase.php');

class user_setsearch_modfunc1 extends t3lib_extobjbase {
    function modMenu() {
        global $LANG;

        return Array (
            'user_setsearch_modfunc1_check' => '',
        );
    }

    function main() {
        global $BE_USER, $LANG, $BACK_PATH;

        $out.=$this->pObj->doc->spacer(5);
        $out.=$this->pObj->doc->section($LANG->getLL('title'),
                                     'Dummy content here...', 0, 1);

        $menu=array();
        $menu[]=t3lib_BEfunc::getFuncCheck($this->pObj->id,
            'SET[user_setsearch_modfunc1_check]',
            $this->pObj->MOD_SETTINGS['user_setsearch_modfunc1_check']).
            $LANG->getLL('checklabel');
        $out.=$this->pObj->doc->spacer(5);
        $out.=$this->pObj->doc->section('Menu',implode(' - ', $menu), 0, 1);

        return $out;
    }
}

```

Une fonction de sous-module est une extension de `t3lib_extobjbase`. Les fonctions de sous-module sont instanciées et initialisées par le module ou par la classe `t3lib_SCbase`. Par conséquent, les fonctions de sous-module ne peuvent normalement contenir que les méthodes `modMenu()` et `main()`. Si la fonction de sous-module n'a pas ses propres options de menu qui doivent être enregistrées dans la configuration du module, alors la méthode `modMenu()` n'est pas nécessaire.

La méthode `main()` est appelée par le module parent et doit renvoyer le code HTML pour la fonction de sous-module. Le code HTML est normalement généré dans les modules backend par l'instance de la classe `template` (`typo3/template.php`), disponible dans le module

via `$this->doc`. Puisque l'objet du module parent est disponible dans la fonction de sous-module via `$this->pObj`, on accède à l'objet `template` par `$this->pObj->doc`. Il s'agit de la différence principale entre les fonctions de sous-modules, les modules et les sous-modules.

La fonction est supposée afficher l'arborescence des pages à modifier. La classe `t3lib_pageTree` déjà disponible est utilisée à cet effet. Mais elle nécessite d'être quelque peu modifiée et étendue. Puisque l'extension, de par sa clé d'extension, a comme espace de nommage `user_setsearch`, la classe qui en dérive est nommée `user_setsearch_pageTree`.

```
require_once(PATH_t3lib.'class.t3lib_pagetree.php');

/**
 * local version of the page tree
 * which points the title link to the current script
 *
 * @author René Fritz <r.fritz@colorcube.de>
 */
class user_setsearch_pageTree extends t3lib_pageTree {
    function wrapTitle($title,$v) {
        $aOnClick = 'return jumpToUrl(\'\'.$this->thisScript.'?id=' .
                                                    $v['uid'].'\'',this);';
        return '<a href="#" onclick="'.htmlspecialchars($aOnClick).'">' .
                                                    $title.'</a>';
    }
}
```

Seule la méthode `wrapTitle()` est modifiée pour créer un lien approprié vers le titre de la page. En cliquant sur le titre de la page, l'utilisateur détermine aussi la page de départ lorsque l'arborescence est affichée. Pour ce faire, le script du module parent est appelé via le paramètre `id` avec l'`uid` de la page comme valeur. Le paramètre `id` est standard pour les modules **Web**, il est transféré par l'arborescence au cadre de navigation. La classe de module `t3lib_SCbase` reconnaît ce paramètre et l'enregistre dans la variable `$this->id`. De cette façon, l'`ID` est aussi disponible dans la fonction de sous-module via `$this->pObj->id`.

La classe de fonction de sous-module, comme nous l'avons déjà mentionné, est une extension de `t3lib_extobjbase`. Les variables requises sont définies dans l'en-tête de la classe.

```
require_once(PATH_t3lib.'class.t3lib_extobjbase.php');

/**
 * Creates the "set searchable" wizard
 *
 * @author René Fritz <r.fritz@colorcube.de>
 */
class user_setsearch_modfunc1 extends t3lib_extobjbase {

    /**
     * Page tree object
     * @see t3lib_pageTree
     */
    var $tree;

    /**
```



```

    * The current target script (index.php)
    */
    var $thisScript;

```

Le menu de sélection déterminant le nombre de niveaux dans l'arborescence des pages est défini dans la méthode `modMenu()` et est renvoyé dans un tableau. Ce dernier est associé au tableau du module `$MOD_MENU` et ses valeurs sont automatiquement enregistrées avec la configuration du module. On a déjà prévu la traduction du mot « levels » qui est utilisé ici.

```

/**
 * Adds menu items: Levels menu
 *
 * @return array
 * @ignore
 */
function modMenu() {
    global $LANG;

    $levelsLabel = $LANG->sL(
        'LLL:EXT:lang/locallang_mod_web_perm.php:levels');
    return array(
        'user_setsearch_modfunc1_depth' => array(
            1 => '1 ' . $levelsLabel,
            2 => '2 ' . $levelsLabel,
            3 => '3 ' . $levelsLabel,
            4 => '4 ' . $levelsLabel,
            10 => '10 ' . $levelsLabel
        )
    );
}

```

La méthode principale est appelée par le module parent et renvoie le code HTML.

```

/**
 * Main function creating the content for the module.
 *
 * @return string HTML content for the module, actually a "section"
 *               made through the parent object in $this->pObj
 */
function main() {
    global $BE_USER, $LANG, $BACK_PATH;

    $this->thisScript = basename(PATH_thisScript);

```

Le nom du script du module parent est enregistré dans `$this->thisScript`, car il sera utilisé à plusieurs reprises.

Ensuite, on appelle `$this->getPageTree()`. Cette méthode, qui sera décrite plus tard, initialise l'arborescence.

```

    $this->getPageTree();

    $out = '';

```

```
// title
$out.= $this->pObj->doc->spacer(5);
$out.= $this->pObj->doc->section($LANG->getLL('title'),' ',0,1);
```

La variable `$out` accumule le résultat HTML. On affiche d'abord le titre de cette fonction de sous-module. Le titre est défini dans le fichier `locallang.php` et est automatiquement lu lors de l'initialisation de la fonction de sous-module.

Le menu de sélection servant à déterminer le nombre de niveaux dans l'arborescence est repris ci-dessous. La fonction `t3lib_BEfunc::getFuncMenu()` crée un menu en se basant sur les paramètres GET/POST de `SET[]`. La classe de base du module `t3lib_SCbase` reconnaît tous les paramètres passés par `SET[]` et les compare avec la configuration du module, disponible dans `MOD_SETTINGS`. De cette façon, le module lui-même ne se soucie pas de l'enregistrement des options du module. Le préfixe correct (`user_setsearch_modfunc1`) doit être de toute manière utilisé pour éviter les collusions avec d'autres extensions.

```
// depth menu
$menu = $LANG->sL('LLL:EXT:lang/locallang_mod_web_perm.php:Depth').
': '.
t3lib_BEfunc::getFuncMenu($this->pObj->id,
    'SET[user_setsearch_modfunc1_depth]',
    $this->pObj->MOD_SETTINGS['user_setsearch_modfunc1_depth'],
    $this->pObj->MOD_MENU['user_setsearch_modfunc1_depth']);

$out.= $this->pObj->doc->spacer(5);
$out.= $this->pObj->doc->section(' ', $menu, 0, 1);
```

Ensuite, l'arborescence créée par la méthode `showPageTree()` est affichée.

```
// output page tree
$out.= $this->pObj->doc->spacer(10);
$out.= $this->pObj->doc->section(' ', $this->showPageTree(), 0, 1);
```

Puisque le module parent a déjà inséré la balise `<form>`, cette dernière doit d'abord être refermée, puisqu'une autre URL est requise pour spécifier l'action de cette fonction de sous-module.

```
// new form (close old)
$out.= '</form>';
$out.= $this->pObj->doc->spacer(10);
```

Dans le sous-module, on reprend seulement les ID des pages dont la case à cocher **Sans recherche** doit être modifiée. La modification de cette valeur est effectuée par le TCE (TYPO3 Core Engine), qui est disponible pour les modules dans le script `typo3/tce_db.php`. Une nouvelle balise `<form>` est insérée avec `tce_db.php` comme cible.

```
// call tce_db.php script with the commands
$out.= '<form action="'.
    $BACK_PATH.'tce_db.php' method="POST" name="editform">';
$out.= '<input type="hidden" name="id" value="'.
    $this->pObj->id.'">';
```

Le script `tce_db.php` accepte une série de commandes qui sont passées, dans les lignes qui suivent, en tant qu'éléments cachés. Tout d'abord, l'URL de redirection est spécifiée par `redirect`. Il s'agit en fait du module parent dont le nom est déjà présent dans `$this->thisScript`. La constante `TYPO3_MOD_PATH` définit le chemin relatif du script à partir du répertoire `typo3/` où se situe `tce_db.php`.

```
$out.= '<input type="hidden" name="redirect" value="'.
      TYPO3_MOD_PATH.$this->thisScript.'?id='.$this->pObj->id.'">';
```

De cette manière, un renvoi vers le script appelant est fait via une redirection, sans que l'utilisateur ne remarque l'appel à `tce_db.php`.

Nous considérons à présent la commande pour modifier la valeur de la case à cocher **Sans recherche**. La première ligne demande à `tce_db.php` de mettre à 1 le champ `no_search` de l'enregistrement de la table `pages` dont l'ID est `$this->pObj->id`. En théorie, on doit répéter cette action pour chaque page de l'arborescence sélectionnée ; mais il existe une façon plus élégante de procéder.

```
$out.= '<input type="hidden" name="data[pages]['.
      $this->pObj->id.'][no_search]" value="1">';
```

Grâce à la commande `mirror`, vous demandez à `tce_db.php` d'effectuer le même changement sur une série d'enregistrements. La méthode `$this->getEditablePagesIDList()`, qui est reprise ci-dessous, fournit les ID des pages de l'arborescence.

```
$out.= '<input type="hidden" name="mirror[pages]['.
      $this->pObj->id.']" value="'. $this->getEditablePagesIDList().'">';
```

On donne un nom aux boutons de soumission suivant leur fonction afin d'envoyer le formulaire. Un bouton a mis, via JavaScript, la valeur de la case à cocher de 1 à 0 dans la balise `<input>`.

```
// submit buttons
$out.= '<input type="submit" name="setSearchable" value="'.
      $LANG->getLL('setSearchable').
      '" onclick="document.editform[\'data[pages]['.
      $this->pObj->id.'][no_search]\'.value=0;"> ';
$out.= '<input type="submit" name="setNonSearchable" value="'.
      $LANG->getLL('setNonSearchable').'">';

return $out;
}
```

Pour terminer, le code HTML est renvoyé au module appelant. À propos, la balise `<form>` n'est pas fermée, puisque le module insérera une ou plusieurs balises pour terminer son propre formulaire. La méthode `getPageTree()` utilise la classe `user_setsearch_pageTree` pour créer l'arborescence. La méthode `$this->pObj->perms_clause` est disponible pour déterminer l'expression SQL qui exclut les pages pour lesquelles l'utilisateur n'a pas de permission.

```
/**
 * Reads the page tree
```

```

*
* @return    void
*/
function getPageTree() {
    global $BE_USER,$LANG,$BACK_PATH;

    $this->tree = t3lib_div::makeInstance('user_setsearch_pageTree');
    $this->tree->init(' AND '.$this->pObj->perms_clause);

```

En spécifiant `setRecs=true`, les enregistrements des pages sont rassemblés par l'objet et sont disponibles par après. `makeHTML=true` active la génération de l'arborescence sous forme de code HTML. Le paramètre `$this->thisScript` est transféré puisque `wrapTitle()` doit créer un hyperlien.

```

    $this->tree->setRecs = true;
    $this->tree->makeHTML = true;
    $this->tree->thisScript = $this->thisScript;

```

On ajoute alors l'ensemble des champs de la base de données qui devront être disponibles par la suite.

```

    $this->tree->addField('no_search');
    $this->tree->addField('perms_userid',1);
    $this->tree->addField('perms_groupid',1);
    $this->tree->addField('perms_user',1);
    $this->tree->addField('perms_group',1);
    $this->tree->addField('perms_everybody',1);
    // Creating top icon; the current page
    $HTML = t3lib_iconWorks::getIconImage('pages', $this->pObj->pageinfo,
                                           $BACK_PATH, 'align="top"');
    $this->tree->tree[] =
        array('row'=>$this->pObj->pageinfo, 'HTML'=>$HTML);

    // read the page data and create the tree
    $this->tree->getTree($this->pObj->id,
        $this->pObj->MOD_SETTINGS['user_setsearch_modfunc1_depth']);
}

```

Avant que l'arborescence ne soit finalement lue par `getTree()`, la racine, c'est-à-dire la page courante, doit être déterminée.

La méthode `showPageTree()` crée la table avec une arborescence et l'affichage du statut. Bien que la classe `t3lib_pageTree` ne contienne pas la méthode permettant d'afficher l'arborescence complète en HTML, on ne l'utilise pas ici, car la case à cocher doit être reprise dans le tableau affiché à côté de la page.

Il est assez normal d'élaborer le code HTML manuellement dans les modules. Mais il existe un certain nombre de fonctions dans `t3lib_BEfunc` ou dans l'objet `template` (`->doc`) qui peuvent vous être utiles. `t3lib_BEfunc::getFuncMenu()` ou `->doc->section()` ont déjà été données comme exemple. Dans cette méthode, le tableau de l'arborescence des pages est créé par `->doc->table()`. Comme nous l'avons déjà dit, vous pouvez placer le tableau directement dans le code.

Tout d'abord, la mise en forme du tableau est définie par le tableau `$tableLayout`. La balise `<table>` est définie, avec `defRow` contenant les balises pour les lignes et les colonnes 0 et 1. Une couleur d'arrière-fond différente est définie pour la ligne 0 dans la balise `<tr>`.

```
/**
 * Creates the page tree table
 *
 * @return string rendered HTML table
 */
function showPageTree() {
    global $BE_USER,$LANG,$BACK_PATH;

    // init table layout
    $tableLayout = array (
        'table' => array ('<table border="0" cellpadding="0" cellspacing="0" id="typo3-tree" style="width:auto;">', '</table>'),
        'defRow' => array (
            'tr' => array ('<tr class="bgColor-20">', '</tr>'),
            '0' => array ('<td nowrap="nowrap">', '</td>'),
            '1' => array ('<td align="center" style="border-left: solid 1px '.$this->pObj->doc->bgColor.'">', '</td>'),
        ),
        '0' => array (
            'tr' => array ('<tr class="bgColor2">', '</tr>'),
        )
    );
};
```

Le tableau bidimensionnel `$table` rassemble les contenus du tableau en lignes et en colonnes. L'icône `tt_content_search` est insérée dans la première ligne. L'icône est normalement utilisée pour le type de contenu **Recherche** et servira à illustrer la colonne indiquant le statut de la case à cocher **Sans recherche**.

```
$table = array();
$str = 0;

// add header row
$table[$str][0] = '&nbsp;';
$table[$str++][1] = '&nbsp;'.t3lib_iconWorks::getIconImage(
    'tt_content_search',
    array(),
    $BACK_PATH). '&nbsp;';

// walk through the tree list
foreach($this->tree->tree as $pageItem) {
```

À présent, chaque élément de l'arborescence déjà constituée est traité dans une boucle `foreach`. Les lignes suivantes spécifient une couleur d'arrière-fond différente pour la ligne si l'utilisateur n'a pas de permission d'édition (2) pour la page.

```
    // if user has no access use a darker row background
    if (!(($this->admin ||
        $BE_USER->doesUserHaveAccess($pageItem['row'],2))) {
        $tableLayout[$str]['tr'] =
```

```

        array('<tr class="bgColor4">', '</tr>');
    }

    // get one page tree item
    $title = t3lib_div::fixed_lgd(
        $this->tree->getTitleStr($pageItem['row']),
        $BE_USER->uc['titleLen']);
    $treeItem = $pageItem['HTML'];
    $this->tree->wrapTitle($title, $pageItem['row']);

```

Lorsque le titre de la page a été créé, et que l'icône existante a été ajoutée, le statut de la case à cocher **Sans recherche** est déterminé : selon la valeur, on utilise le rouge ou le vert.

```

    // get current no_search flag
    if ($pageItem['row']['no_search']) {
        $searchFlag = '<span style="color:red">&times;</span>';
    } else {
        $searchFlag = '<span style="color:green">&bull;</span>';
    }

    // add row to table
    $table[$tr][0] = $treeItem.'&nbsp;&nbsp; ';
    $table[$tr++][1] = $searchFlag;
}

// return rendered table
return $this->pObj->doc->table($table, $tableLayout);
}

```

Ensuite, la ligne du tableau est ajoutée, et à la fin de la boucle, le tableau est créé par `->doc->table()` et renvoyé.

En utilisant la classe `t3lib_pageTree` (voir aussi `t3lib_browseTree`), vous pouvez vous faciliter la tâche si vous devez manipuler des arborescences dans vos propres modules. D'une part, la classe fournit tous les enregistrements et, d'autre part, elle donne, si nécessaire, l'arborescence pour l'affichage.

Il manque encore la méthode `getEditablePagesIDList()`, qui retourne une liste d'ID de pages séparés par des virgules, pour lesquelles l'utilisateur a les droits d'édition. Cette liste est dressée dans le formulaire dans la méthode `main()` et transférée à `tce_db.php`.

```

/**
 * Returns a comma separated list of page id's
 * which are accessible to the user
 *
 * @return string pages uid list
 */
function getEditablePagesIDList() {
    global $BE_USER, $LANG, $BACK_PATH;

    $idListArr=array();

    foreach ($this->tree->tree as $pageItem) {
        if ($this->admin ||
            $BE_USER->doesUserHaveAccess($pageItem['row'],2)) {

```

```

        $idListArr[] = $pageItem['row']['uid'];
    }
}
return implode(',', $idListArr);
}
}

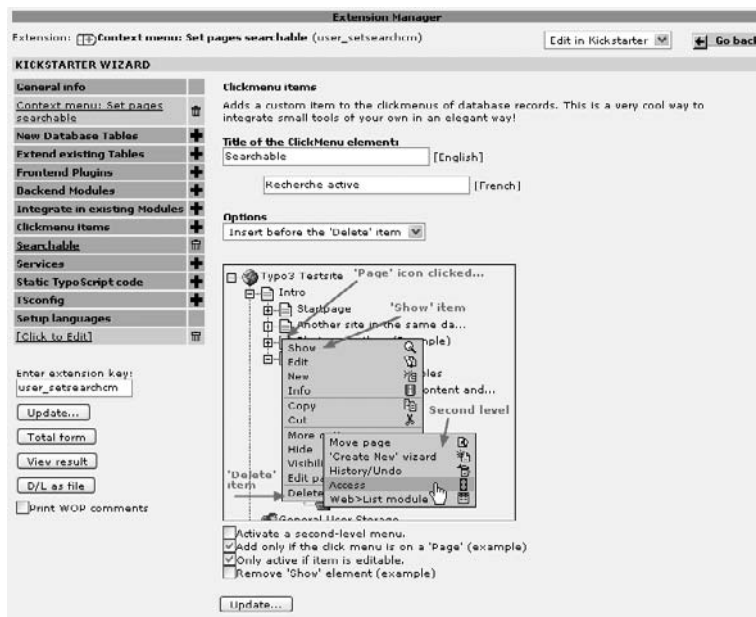
```

7.8.3 Menu contextuel

Dans cet exemple, un menu contextuel doit remplir le même rôle que les fonctions précédentes de sous-modules, c'est-à-dire permettre ou non la recherche sur la page. Mais cette fois-ci, on ne considère que la page sélectionnée.

On crée rapidement une extension dans le Kickstarter pour insérer un élément dans le menu contextuel. Comme toujours, l'extension est constituée par le composant **General info**, par les langues supplémentaires et par les composants pour le menu contextuel, qui est créé par **Clickmenu items**.

Figure 7.45:
Création d'un menu
contextuel par le
Kickstarter



Le Kickstarter crée les fichiers suivants :

```

ext_emconf.php
ext_icon.gif
ext_tables.php
class.user_setsearchcm_cm1.php
locallang.php
cm1/
    clear.gif
    cm_icon.gif

```

```

conf.php
index.php
locallang.php

```

Si vous installez et testez cette extension, une nouvelle entrée apparaîtra dans le menu contextuel, comme prévu. Un exemple de module créé par le Kickstarter est appelé. Il est situé dans le répertoire `cm1/`.

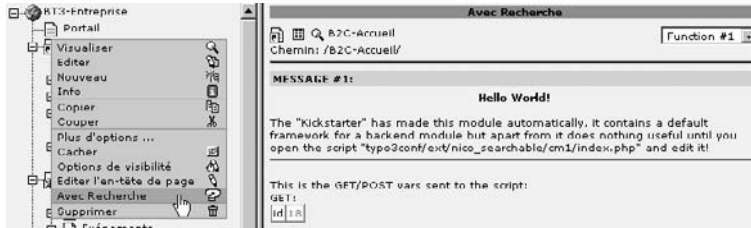


Figure 7.46:
Nouvelle entrée
affichant le module

L'extension `Extra Click Menu Options` (`extra_page_cm_options`) propose pour l'essentiel la fonctionnalité d'activation ou non de la recherche dans une page. Si vous examinez la manière dont cette fonction est implémentée, vous arriverez à la méthode `DB_changeFlag()`, qui se trouve dans `typo3/alt_clickmenu.php`.

La méthode fournit déjà la fonctionnalité souhaitée pour changer la valeur du champ de la base de données (0 ou 1). La méthode est relativement dense, et il n'est pas nécessaire de la comprendre complètement à ce stade. Mais il est intéressant de noter que le même concept est à nouveau utilisé, consistant à appeler le TCE via le script `tce_db.php`, pour apporter les modifications à l'enregistrement de la page. Cela signifie que votre propre module, dans `cm1/`, est inutile et peut être effacé.

Les icônes représentant les boutons dans le menu contextuel sont ajoutées à l'extension, ce qui explique la présence des fichiers suivants :

```

button_no_search.gif
button_unno_search.gif
ext_emconf.php
ext_icon.gif
ext_tables.php
class.user_setsearchcm_cm1.php
locallang.php

```

Le fichier `class.user_setsearchcm_cm1.php` contient la classe qui génère la nouvelle entrée du menu. Le code a été créé en grande partie par le Kickstarter. Nous ne mentionnerons ici que ses caractéristiques particulières.

```

class user_setsearchcm_cm1 {

/**
 * Adding options to the context menu.
 *
 * @param object      The click menu object
 * @param array       Menu items array

```



```

* @param    string      Name of the table of the clicked record item
* @param    integer     uid of the record
* @return   array       Menu items array, processed.
*/
function main(&$cmObj, $menuItems, $table, $uid) {
    global $LANG;

    if (!$cmObj->cmLevel) {
        if ($cmObj->editOK) {

            // Returns directly,
            // because the clicked item was not from the pages table
            if ($table != 'pages')    return $menuItems;

            // load the language array
            $LL = $this->includeLL();

            // array for new menu items
            $localItems = array();

            // create new menu item if not disabled
            if (!in_array('user_setsearchcm_cml',
                        $cmObj->disabledItems)) {

```

On vérifie si l'entrée du menu a été désactivée, ce qui est possible, par exemple, via TSConfig.

```

$flagField = 'no_search';
$title = ($cmObj->rec[$flagField]) ?
    $LANG->getLLL('searchable', $LL) :
    $LANG->getLLL('non_searchable', $LL);

$localItems['user_setsearchcm_cml'] =
    $cmObj->DB_changeFlag(
        'pages',
        $cmObj->rec,
        $flagField,
        $title,
        'no_search',
        t3lib_extMgm::extRelPath('user_setsearchcm'));

```

Grâce à la méthode `DB_changeFlag()` de l'objet de menu contextuel, on crée une entrée pour le champ `no_search` de la table `pages`, qui selon son statut, affiche une des icônes `button_*` et contient un lien vers le script `tce_db.php`, pour modifier la valeur du champ à 0 ou 1.

```

// add menu item
$menuItems = $cmObj->addMenuItems(
    $menuItems,
    $localItems,
    'after:hide,before-spacer:delete');

```

L'entrée du menu créé est insérée par la méthode `addMenuItems()` à la bonne place dans le menu existant, c'est-à-dire après `cache` (si celui-ci est disponible) ou avant `Supprimer` (y compris la ligne de séparation).

```

    }
  }
  return $menuItems;
}

```

Pour finir, le tableau de menu modifié est renvoyé.

```

/**
 * Includes the [extDir]/locallang.php and
 * returns the $LOCAL_LANG array found in that file.
 *
 * @return array $LOCAL_LANG array
 */
function includeLL() {
    include(t3lib_extMgm::extPath('user_setsearchcm'). 'locallang.php');
    return $LOCAL_LANG;
}
}

```

De plus, la classe contient à présent une seule méthode de lecture du fichier des langues.

Bien sûr, de nombreuses entrées peuvent être insérées par la méthode présentée plus haut. De même, on peut aussi créer de nouvelles entrées dans le second niveau de menu. L'extension Extra Click Menu Options (extra_page_cm_options) peut ici servir d'exemple.

7.8.4 Habillages – Changer l'apparence du backend

L'apparence du backend peut être modifiée dans une large mesure. C'est là une opération facile à réaliser parce qu'il existe une interface permettant de changer à la fois les couleurs, les logos et les icônes du backend. Mais il y a des limites : les mentions de copyright et de la licence GPL, de même que les scripts, ne peuvent pas être modifiés.

Référence 840654

Ces restrictions mises à part, vous pouvez certainement utiliser votre propre logo ou celui de vos clients à la place des images, comme le montre l'exemple qui suit.



Figure 7.47:
Habillage pour le livre

Comme d'habitude, les modifications sont apportées via des extensions. Voici une liste des fichiers pour cet exemple d'habillage. Les fichiers comportant des modifications pour l'identification sont repris en gras.

```
ext_tables.php
stylesheet_post.css
backgrounds/
    csm_back.png
    login_back.jpg
    logoframe_back.png
    mainmenu_back.png
    menu_back.jpg
    topframe_back.png
icons/
    gfx/
        alt_backend_logo.png
        altmenuline.png
        typo3logo.png
    fileicons/
        ai.png
        au.png
        avi.png
        ...
    i/
        _icon_folders.png
        ...
        tt_content.png
        ...
loginimages/
    01.png
    02.png
```

Comme vous pouvez le remarquer, il y a ici un seul fichier d'extension : `ext_tables.php`. L'habillage y est déclaré au système. Tous les autres fichiers sont des feuilles de style, des éléments graphiques, ou des icônes.

`ext_tables.php`

Tout d'abord, le chemin d'extension relatif au répertoire du backend (`typo3/`) est déterminé et sauvé dans une variable temporaire. Cela évite par la suite plusieurs appels à la fonction `t3lib_extMgm::extRelPath()`.

```
// Setting the relative path to the extension in temp. variable:
$temp_eP = t3lib_extMgm::extRelPath($_EXTKEY);
```

Toutes les modifications sont reportées à TYPO3 via le tableau global `$TBE_STYLES`. Pour la page d'identification, on spécifie le chemin où sont situées les images illustratives à placer sur la partie gauche de la page. Si plusieurs images sont disponibles, elles seront sélectionnées au hasard.

```
// Setting login box image rotation folder:
$TBE_STYLES['loginBoxImage_rotationFolder'] = $temp_eP.'loginimages/';
```

Tous les fichiers graphiques de TYPO3 sont situés dans le répertoire `t3lib/gfx/`. Pour remplacer les images et les icônes, vous devez indiquer à TYPO3 le répertoire où se trouvent les

nouvelles images. Si vous mettez en place la structure du répertoire `gfx/` dans le nouveau répertoire d'images, les fichiers remplaceront les fichiers originaux de même nom se trouvant dans `t3lib/gfx/`. Cela vaut aussi pour `gfx/typo3logo.gif` qui représente le logo s'affichant en haut à droite de la page d'identification.

```
// Setting up auto detection of alternative icons:
$TBE_STYLES['skinImgAutoCfg']=array(
    'absDir' => t3lib_extMgm::extPath($_EXTKEY).'icons/',
    'relDir' => $temp_eP.'icons/',
    // Force to look for PNG alternatives...
    'forceFileExtension' => 'png',
);
```

Les autres modifications sont faites via des feuilles de style.

```
// Additional stylesheet. Set AFTER any styles in the document
$TBE_STYLES['stylesheetFile_post'] = $temp_eP.'stylesheet_post.css';
```

stylesheet_post.css

```
/* Login Screen */
BODY#typo3-index-php { background-color: #fff; }
BODY#typo3-index-php TABLE#loginwrapper { background-color: #C2C9CD; }
BODY#typo3-index-php DIV#copyrightnotice { font-size: 11px;}
TABLE#logintable INPUT { border: #7B8295 solid 1px; }
```

Comme nous l'avons déjà mentionné, ces possibilités ne se limitent pas à l'identification. Vous pouvez changer complètement l'apparence du backend, comme le montre la figure suivante. Un exemple de refonte du backend est l'extension d'habillage `skin360` (voir la référence de cette section).

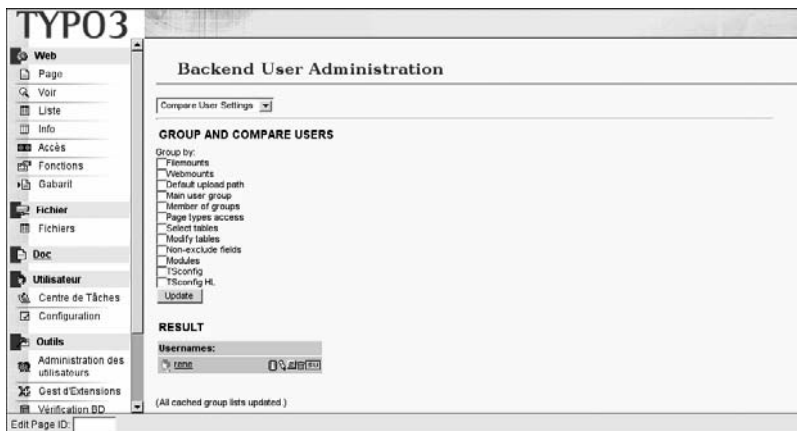


Figure 7.48:
L'habillage « livre de
TYPO3 » dans le
backend

7.9 Services

Référence 052294

Les services fournissent un moyen de développer des fonctionnalités dans TYPO3 qui pourront être étendues ou remplacées.

L'extension DAM utilise des services, par exemple, pour lire les données méta des fichiers. Chaque format de fichier sauvegarde les données méta différemment (pour autant que cela soit permis). Il est donc nécessaire d'écrire une fonction ou une classe particulière pour chaque type de fichier pour lire ses données, les traiter et les convertir dans un format uniforme. Le DAM devrait alors appeler la fonction concernée, selon le type de fichier. Cette procédure n'est pas très flexible, bien sûr. Les nouvelles fonctions, et par conséquent les types de fichiers correspondants, ne seraient disponibles qu'après une mise à jour du DAM. De plus, les tiers auraient peu de possibilités pour de nouveaux développements. Les services offrent un niveau d'abstraction qui contourne ces désavantages.

Grâce aux services, vous êtes en position d'utiliser les classes PHP sans connaître leur nom. Normalement, la création d'une classe dans TYPO3 ressemble à ceci :

```
require_once(t3lib_extMgm::extPath('some_extension').
    'class.tx_some_extension_class.php');
$obj = t3lib_div::makeInstance('tx_some_extension_class');
```

Pour créer un objet, vous devez connaître le fichier PHP dans lequel se situe la classe, ainsi que le nom de la classe. Ce n'est pas nécessaire lorsqu'on recourt aux services. Un objet est créé de la manière suivante :

```
$serviceObj = t3lib_div::makeInstanceService('my_service_type');
```

La différence, c'est que la classe est automatiquement sélectionnée et que l'objet correspondant est créé sur base du type de service spécifié. Les services doivent donc avoir un type. L'extension DAM utilise par exemple les services de type `metaExtract` pour lire des données méta.

Les services existent pour deux raisons :

1. *Liberté de mise en œuvre*

Un service peut être installé de différentes façons, à plusieurs reprises, et est donc remplaçable. C'est un avantage si la fonctionnalité est disponible uniquement sur une plate-forme spécifique, ou via des outils externes. Le service qui est disponible et actif est utilisé automatiquement.

2. *Ajout de fonctionnalités via des extensions*

Chaque service a un type défini par une chaîne de caractères, comparable à la clé d'extension. `metaExtract` est par exemple un type de service. Chaque type de service a sa propre API : un service qui lit les données méta comprendra des méthodes différentes par rapport à un service qui écrit des données de log.

7.9.1 Mise en œuvre des services

Pour utiliser un service, vous devez bien sûr connaître le type de service et son API. Voici un exemple simple :

```

if (is_object($serviceObj =
    t3lib_div::makeInstanceService('textLang')) {
    $language = $serviceObj->guessLanguage($text);
}

```

On sollicite un objet du type de service `textLang` et dans le même temps, on vérifie, via `is_object()`, si un objet a bien été renvoyé. Cette validation est nécessaire parce qu'il existe plusieurs raisons pour lesquelles un service serait indisponible :

- Un service du type souhaité n'est pas installé.
- Le service s'est désactivé lui-même durant l'enregistrement sur le système parce que, par exemple, il ne peut pas fonctionner sur le système.
- Le service a été désactivé par le système suite à une vérification.
- Durant son installation, le service a lui-même vérifié s'il pouvait être opérationnel, et s'est désactivé de lui-même.

Évidemment, il est possible que le service `textLang` ne soit pas installé sur le système. Votre application détermine si le service est obligatoire ou non à son bon fonctionnement. Si nécessaire, vous pouvez afficher un message d'erreur, indiquant à l'utilisateur que le service correspondant doit être installé.

Sous-types

Les services peuvent être sélectionnés, non seulement sur base d'un type, mais aussi sur base d'un sous-type.

```

$absFile = '/tmp/testfile.pdf';
$fileType = 'pdf';

$meta = array();
if (is_object($serviceObj = t3lib_div::makeInstanceService('metaExtract',
    $fileType))) {
    $serviceObj->setInputFile($absFile, $fileType);
    if ($serviceObj->process('', '', $meta) > 0
        AND (is_array($svmeta = $serviceObj->getOutput()))) {
        $meta = $svmeta;
    }
}

```

Ici, un service de type `metaExtract` est sélectionné. Ce service peut traiter le sous-type `pdf`. De cette façon, vous établissez une bibliothèque de services qui ont en commun la même fonctionnalité et la même API (par exemple, pour lire les données méta), mais qui sont implémentées différemment pour chaque sous-type. Les sous-types ne se limitent pas au fichier repris dans cet exemple. Ils peuvent, et doivent, être définis pour chaque service qui les contient. Il va de soi qu'il existe des services sans sous-type.

Services en cascade

Jusqu'à présent, un seul type de service a été créé dans chaque instance pour éditer les données. Mais il peut être utile d'appliquer tous les services disponibles d'un certain type à vos données, ou simplement d'essayer tous les services jusqu'au moment où vous obtenez un résultat.

```
$subType = 'getUserFE';
$serviceChain = array();
while (is_object($serviceObj = t3lib_div::makeInstanceService('auth',
    $subType, $serviceChain))) {

    // add service key to list of tried services
    $serviceChain[] = $serviceObj->getServiceKey();

    // initialize service
    $serviceObj->initAuth($subType, $loginData, $info, $this);

    // call the service to get a login user
    if ($tempuser = $serviceObj->getUser()) {
        // user found, just stop to search
        break;
    }
}
```

Le code est un extrait d'un développement pour utiliser les services d'identification des utilisateurs.

On peut passer au troisième paramètre de `makeInstanceService()` une liste de clés de service (sous forme de tableau ou de liste d'éléments séparés par des virgules). Les services dans cette liste sont par la suite ignorés. De cette manière, vous pouvez appeler les services d'un type les uns après les autres. Dans notre exemple, la boucle s'arrête lorsqu'un utilisateur a été trouvé.

Appel de services spécifiques

Parfois, il peut être utile de travailler sans niveau d'abstraction, et ne pas devoir sélectionner automatiquement un service à partir de `makeInstanceService()`, mais appeler un service en particulier. C'est possible, car un service est enregistré non seulement via son type, mais aussi via la clé de service, qui joue un rôle analogue à celui de la clé d'extension.

```
$serviceObj = t3lib_div::makeInstanceService('textExtract')

$serviceObj = t3lib_div::makeInstanceService('tx_cctxttextphp_sv1')
```

Alors qu'un type de service est spécifié par `textExtract` pour lancer une recherche sur le service correspondant, dans le second exemple, un service spécifique est créé via la clé `tx_cctxttextphp_sv1`.

7.9.2 Développer des services

Le développement d'un service se fait via une extension. Comme d'habitude, nous démarrons dans le Kickstarter. Il est tout à fait possible d'ajouter, au lieu d'un service, un plugin ou un

module à une extension. Mais cela n'a pas beaucoup de sens, puisque pour utiliser ce service, le plugin qui le contient doit être installé, ce qui n'est sans doute pas la meilleure idée.

Choisissez **Services** comme catégorie de l'extension. Le nom du type de service devrait commencer par le titre de l'extension. De cette manière, vous reconnaissez directement quel type de service est inclus. De plus, puisque les extensions sont automatiquement triées par le gestionnaire d'extensions, vous avez une vue d'ensemble. Un nouveau service est ajouté à l'extension avec l'élément de menu **Services**.

The screenshot shows the 'Extension Manager' window with the 'Kickstarter Wizard' active. The 'Menu' dropdown is set to 'Make new extension'. The wizard has a left sidebar with categories: General info, New Database Tables, Extend existing Tables, Frontend Plugins, Backend Modules, Integrate in existing Modules, Clickmenu items, Services, Text extraction for rtf, Text extraction for txt, html, sgml, xml, Static TypoScript code, TSconfig, and Setup languages. The 'Services' category is selected. The main area contains the following fields and instructions:

- Services**: Create a Services class. With a Services extension you can extend TYPO3 (or an extension which use Services) with functionality, without any changes to the code which use that service.
- Title:** Text extraction for rtf
- Description:** This service depends on PHP only.
- Service type:** textExtract
- Sub type(s) (comma list):** rtf
- Priority:** default (50)
- Quality:** 50
- Operating System dependency:** no special dependency
- External program(s) (comma list):**

At the bottom, there is an 'Update...' button.

Figure 7.49:
Formulaire de
définition d'un service
dans le Kickstarter

Les détails suivants doivent être saisis dans le formulaire :

- Title**
Un titre court et descriptif
- Description**
Courte description de la fonction
- Service type**
Type de service, défini par une chaîne de caractères courte ; exemples : metaExtract, textLang

Sub type

Liste des sous-types possibles séparés par des virgules ; ces derniers sont définis par le type de service. Certains types n'ont aucun sous-type. Le service `metaExtract` définit des formats de fichiers (`jpg`, `sxw`, `pdf`, ...) comme sous-types possibles.

Priority

On détermine ici la priorité. Puisque les services sont sélectionnés automatiquement, un mécanisme doit décider quel service sera sélectionné s'il en existe plusieurs. C'est précisément le rôle de cette valeur. La valeur normale est 50, elle doit être comprise dans un intervalle de 0 à 100. Des valeurs au-delà de 100 peuvent être spécifiées par après par l'administrateur pour donner une priorité à des services spécifiques. Si vous voulez des services en cascade, cette valeur détermine l'ordre dans lequel ils sont parcourus. Dans la plupart des cas, il n'est pas nécessaire de modifier la valeur.

Quality

Une autre valeur spécifiant quel service a priorité ; pour le service `texLang`, qui détermine la langue du texte, **Quality** définit le nombre de langues qui sont reconnues par `textLang`. Si deux services `textLang` sont installés avec **Priority**=50, le service qui reconnaît le plus de langues sera choisi. Cette valeur dépend donc de la définition du type de service.

Operating system dependency

Définit si le service peut fonctionner sur des systèmes Windows ou UNIX.

External programs

Nom d'un programme externe utilisé par ce service ; celui-ci devrait normalement être spécifié sans chemin, puisque le système tente de trouver le programme requis par lui-même.

Les fichiers suivants sont créés par le Kickstarter à partir de l'exemple ci-dessus :

```
doc/
  wizard_form.dat
  wizard_form.html
sv1/
  class.tx_cctxttextphp_sv1.php
sv2/
  class.tx_cctxttextphp_sv2.php
ext_emconf.php
ext_icon.gif
ext_tables.php
```

Comme vous le remarquez, cette extension contient deux services dans les répertoires `sv1/` et `sv2/`. Une même extension ne contiendra bien sûr que des services du même type.

Le service est enregistré dans le système par la fonction `t3lib_extMgm::addService()`.

```
t3lib_extMgm::addService($_EXTKEY, 'textExtract' /* sv type */,
'tx_cctxttextphp_sv1' /* sv key */,
    array(

        'title' => 'Text extraction for rtf',
```

```

        'description' => 'This service depends on PHP only.',

        'subtype' => 'rtf',

        'available' => true,
        'priority' => 50,
        'quality' => 50,

        'os' => '',
        'exec' => '',

        'classFile' => t3lib_extMgm::extPath($_EXTKEY) .
'sv1/class.tx_cctxttextphp_sv1.php',
        'className' => 'tx_cctxttextphp_sv1',
    )
);

```

Cela correspond, pour l'essentiel, aux détails du formulaire du Kickstarter.

Par défaut, un service enregistré est disponible si la valeur `available` de la configuration est mise à `true`. Vous pouvez bien sûr spécifier ici une expression booléenne. Si un service dépend par exemple de la fonction PHP `exif_read_data()`, disponible seulement à partir de la version 4.2.0 de PHP, on peut effectuer une vérification grâce à la ligne suivante :

```

        'available' => function_exists('exif_read_data'),

```

Le service est alors automatiquement désactivé si cette fonction n'est pas disponible.

Les classes de service générées par le Kickstarter étendent la classe de base `t3lib_svbase` et contiennent par défaut les méthodes `init()` et `process()`. La méthode `init()` est appelée par le système. Grâce à elle, vous initialisez l'objet et vous vérifiez si le service est vraiment disponible. Si c'est le cas, la valeur `true` est retournée. Si d'autres tests ne sont pas nécessaires, la méthode peut être laissée en l'état, puisqu'elle est incluse dans la classe de base.

L'exemple du service `textExtract` est repris. Ce dernier extrait du texte des documents Word et Excel. La mise en forme est ignorée.

Deux services sont enregistrés dans `ext_tables.php`.

```

t3lib_extMgm::addService($_EXTKEY, 'textExtract' /* sv type */,
'tx_cctxttextexec_sv2a' /* sv key */,
    array(
        'title' => 'Text extraction for Word documents (doc)',
        'description' => 'This service depends on catdoc',

        'subtype' => 'doc,doc',

        'available' => TRUE,
        'priority' => 50,
        'quality' => 50,

        'os' => '',
        'exec' => 'catdoc',
    )
);

```

```

        'classFile' => t3lib_extMgm::extPath($_EXTKEY) .
        'sv2/class.tx_cctxttextexec_sv2.php',
        'className' => 'tx_cctxttextexec_sv2',
    )
);

t3lib_extMgm::addService($_EXTKEY, 'textExtract' /* sv type */,
'tx_cctxttextexec_sv2b' /* sv key */,
    array(
        'title' => 'Text extraction for Excel documents (xls)',
        'description' => 'This service depends on xls2csv',

        'subtype' => 'xls,xlt,xlw',

        'available' => TRUE,
        'priority' => 50,
        'quality' => 50,

        'os' => '',
        'exec' => 'xls2csv',

        'classFile' => t3lib_extMgm::extPath($_EXTKEY) .
        'sv2/class.tx_cctxttextexec_sv2.php',
        'className' => 'tx_cctxttextexec_sv2',
    )
);

```

Si vous observez la définition des clés du service (`tx_cctxttextexec_sv2x`) et la classe spécifiée, vous remarquerez que deux services sont enregistrés avec la même classe. C'est tout à fait permis. Puisque l'implémentation est totalement identique pour les deux services, et puisqu'ils sont destinés aux mêmes types de fichiers, ils ont été développés en une classe. Les types de fichiers `doc` ou `dot` pour les documents Word et `xls`, `xlt` et `xlw` pour les documents Excel ont été spécifiés en tant que sous-types. La variable `exec` contient les noms de fichiers (`catdoc`, `xls2csv`) des outils externes requis pour ce service. Dans la méthode `init()` de la classe de base, on vérifie si ces programmes sont disponibles.

Comme vous le voyez, la classe `tx_cctxttextexec_sv2` ne contient que la méthode `process()`. Le nom `process()` est réservé par convention. Un service contenant cette méthode agit comme un filtre : il peut traiter des données transférées par la variable `$content`, et également des fichiers. L'application appelante peut, de la même manière, récupérer les données en tant que variables ou fichiers. On spécifie la méthode `process()` pour tester de la même manière les services « filtres ». Si vous introduisez un nouveau type de service, vous pouvez utiliser votre propre API et vous ne devez pas passer par la méthode `process()`.

```

require_once(PATH_t3lib.'class.t3lib_svbase.php');

class tx_cctxttextexec_sv2 extends t3lib_svbase {
    var $prefixId = 'tx_cctxttextexec_sv2';
    var $scriptRelPath = 'sv2/class.tx_cctxttextexec_sv2.php';
    var $extKey = 'cc_txttextexec';

```

```

/**
 * performs the text extraction
 *
 * @param   string      Content which should be processed to
 *                      extract text.
 * @param   string      Content type 'doc', 'dot', ...
 * @param   array       Configuration array
 * @return  boolean
 */
function process($content='', $type='', $conf=array()) {

    $this->out = '';

    if ($content) {
        $this->setInput ($content, $type);
    }

    if($inputFile = $this->getInputFile()) {
        switch ($this->inputType) {

            case 'doc':
            case 'dot':
                $cmd = t3lib_exec::getCommand('catdoc').' -d8859-1
"'.$inputFile.'";
                $this->out = shell_exec($cmd);
                break;
            case 'xls':
            case 'xlt':
            case 'xlw':
                $cmd = t3lib_exec::getCommand('xls2csv').' -d8859-1
"'.$inputFile.'";
                $this->out = shell_exec($cmd);
                break;

            // if that is reached the caller made a mistake
            default:
                $this->errorPush(T3_ERR_SV_WRONG_SUBTYPE,
                    'Subtype "'.$this->inputType.'" is not
supported.');
```

```

                break;
            }
        } else {
            $this->errorPush(T3_ERR_SV_NO_INPUT, 'No or empty input.');
```

```

        }

        if ($this->out AND intval($conf['limitOutput'])) {
            $this->out = substr($this->out, 0,
                intval($conf['limitOutput']));
        }

        return $this->getLastError();
    }
}

```

Plusieurs fonctions sont disponibles dans la classe de base pour configurer un filtre. Par exemple, des fichiers temporaires sont créés automatiquement, ou bien des fichiers sont lus dans une variable, en fonction de la requête du service ou de l'application appelante.

Dans cet exemple, les données transférées sont enregistrées, si nécessaire :

```
if ($content) {
    $this->setInput ($content, $type);
}
```

Puisque les programmes externes `catdoc` et `xls2csv` exigent un fichier à traiter, on spécifie à la ligne suivante un fichier avec des données. Ce dernier sera créé automatiquement s'il n'existe pas, parce que les données ont été enregistrées via `setInputFile()`. Si l'application appelante a enregistré précédemment un fichier avec `setInputFile()`, ce dernier est utilisé directement. L'application appelante peut solliciter une variable avec `getOutput()` ou un fichier avec `getOutputFile()`. L'appel au service `textExtract` ressemble à ceci :

```
if (is_object($serviceObj = t3lib_div::makeInstanceService('textExtract',
                                                            $file_type))) {

    $serviceObj->setInputFile($absFile, $file_type);
    $serviceObj->process('', '', $conf);
    $output = $serviceObj->getOutput();

    $serviceObj->unlinkTempFiles();
}
```

L'appel de la méthode `unlinkTempFiles()` est important pour supprimer les fichiers temporaires du service. Sinon, vous pouvez, grâce à la fonction

```
... = &t3lib_div::makeInstanceService( ...
```

demander une référence à l'objet, qui supprimera indépendamment les fichiers temporaires.

7.9.3 Configuration

Les données de configuration pour les services sont stockées dans le tableau `$TYPO3_CONF_VARS['SVCONF'][serviceType]`.

```
$TYPO3_CONF_VARS['SVCONF'][serviceType]['setup']
```

Ce tableau peut contenir les données de configuration du script appelant, et n'est pas pris en compte par les services eux-mêmes.

```
$TYPO3_CONF_VARS['SVCONF'][serviceType]['default']
```

Les valeurs par défaut pour le type de service sont définies dans ce tableau. Ces dernières sont utilisées tant que des valeurs particulières ne sont pas définies par la clé de service.

```
$TYPO3_CONF_VARS['SVCONF'][serviceType][serviceKey]
```

La classe de base des services `t3lib_svbase` fournit une méthode `getServiceConfig()` pour lire les valeurs de configuration à partir des tableaux présentés ci-dessus, où la configuration avec les clés de service a priorité sur la configuration par défaut.

7.9.4 Introduction d'un nouveau type de service

Les différents types de services ne sont pas figés. N'importe qui peut introduire un nouveau type de service. Si vous souhaitez aller dans cette direction, l'API destinée au nouveau type doit être discutée avec d'autres développeurs, puisque utiliser un service une seule fois n'a pas beaucoup de sens. Vous devriez introduire le nouveau service dans la liste de diffusion des développeurs TYPO3, et si possible inclure un exemple d'implémentation à télécharger. La documentation de l'API et des sous-types est généralement mise à disposition par la suite.

7.10 XCLASS : modification et extension de classe

Comme nous l'avons vu, TYPO3 peut être étendu de manière simple grâce au système d'extensions. Les fonctionnalités sont encapsulées dans des plugins ou des modules qui sont facilement installés. Mais pour des applications plus complexes, il peut être nécessaire d'apporter des changements à TYPO3 lui-même. Cela ne pose pas de problème, bien sûr, puisque le code source de TYPO3 est disponible. Mais cela rend plus difficile la mise à jour en cas de nouvelle version de TYPO3. Les changements doivent être documentés pour pouvoir être ajoutés plus tard dans la nouvelle version.

Référence 745255

TYPO3 propose les XCLASS comme solution élégante à ce problème. Le concept des XCLASS permet de modifier ou d'étendre presque toutes les classes de TYPO3.

Supposons que vous ayez besoin d'une fonction dans l'objet TypoScript `stdWrap` pour convertir un entier en mots. Cette fonction sera alors disponible pour tout le système, et vous pourrez l'utiliser partout où l'objet TypoScript de type `stdWrap` est défini.

On trouve rapidement sur Internet un script approprié de conversion de nombre. La seule question est à présent de savoir comment l'intégrer au mieux dans la fonction `stdWrap`. Il existe ici plusieurs possibilités :

1. Intégrer la fonctionnalité directement dans la fonction `stdWrap()` dans la classe `class.tslib_content.php`, avec le désavantage déjà mentionné : les mises à jour sont plus difficiles.
2. `stdWrap` fournit un moyen d'intégrer des fonctions utilisateurs. C'est en soi une possibilité valable et utile. Mais, dans notre exemple, nous voudrions faire directement référence à la fonction avec l'identifiant `userNumToWord`.
3. Extension de la méthode `stdWrap` dans la classe `tslib_content` via XCLASS. Cette méthode est celle qui répond le mieux à nos besoins.

Si nous observons la fin du fichier `class.tslib_content.php`, nous avons les lignes suivantes :

```
if (defined('TYPO3_MODE') && $TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS'] [
    'tslib/class.tslib_content.php']) {
    include_once($TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS'] [
        'tslib/class.tslib_content.php']);
}
```

Par conséquent, si la variable

```
$TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS']['tslib/class.tslib_content.php']
```

contient le nom d'un fichier, ce fichier sera inséré par `include_once()`. Cela signifie que nous pouvons insérer un fichier qui peut alors étendre la classe `tslib_content`. Presque tous les fichiers de TYPO3 contenant une classe, même ceux des extensions, comprennent ces lignes relatives à XCLASS dans les dernières lignes.

À présent, il vous reste à initialiser la variable présentée ci-dessus en fonction de vos besoins, et votre script sera intégré. Cette opération est réalisée dans le fichier `ext_localconf.php` de votre propre extension. Gardez à l'esprit que `TYPO3_MODE` est une constante, définie soit comme 'FE', soit comme 'BE' selon que vous êtes dans le frontend ou dans le backend. Dans notre exemple frontend, nous devons donc remplacer `TYPO3_MODE` par 'FE'.

L'exemple d'extension avec la clé `user_NumToWord` ressemble à ceci :

```
doc/  
ext_emconf.php  
ext_icon.gif  
ext_localconf.php  
class.ux_tslib_content.php
```

`ext_localconf.php`

```
if (!defined('TYPO3_MODE'))    die('Access denied.');
```



```
$TYPO3_CONF_VARS['FE']['XCLASS']['tslib/class.tslib_content.php']  
=t3lib_extMgm::extPath('user_NumToWord').'class.ux_tslib_content.php';
```

`class.ux_tslib_content.php`

```
class ux_tslib_cObj extends tslib_cObj {  
  
    function stdWrap($content,$conf) {  
        // Call the real stdWrap function in the parent class:  
        $content = parent::stdWrap($content,$conf);  
  
        // Process according to my user-defined property:  
        if ($conf['userNumToWord']) {  
            $content = $this->ux_numToWord($content,  
                                           $conf['userNumToWord .']);  
        }  
        return $content;  
    }  
  
    function ux_numToWord($content,$conf) {  
        ...  
        return $content;  
    }  
}  
  
if (defined('TYPO3_MODE') && $TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS'] [  
'ext/user_NumToWord/class.ux_tslib_content.php']) {  
    include_once($TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS']
```

```
[ 'ext/user_NumToWord/class.ux_tslib_content.php' ] );
}
```

L'extension via XCLASS est terminée. Cependant, si vous avez été attentif, vous pourriez vous demander comment le système sait que `ux_tslib_cObj` doit être instancié à la place de `tslib_cObj`. C'est grâce à `t3lib_div::makeInstance()`, qui est utilisé au sein de TYPO3 à la place de `new()`.

Les dernières lignes du fichier `class.ux_tslib_content.php` contiennent le code XCLASS habituel. Il est donc aussi possible d'étendre la classe `ux_tslib_cObj`.

La méthode XCLASS comporte aussi des désavantages : d'une part, seule une XCLASS est possible par fichier, et d'autre part, vous n'êtes pas tout à fait à l'abri de tout problème lors d'une mise à jour de TYPO3, puisque l'API de la classe étendue peut avoir changé. Mais cela reste un moyen utile pour apporter des changements au cœur du code source.

7.11 TYPO3 et autres langages de programmation

D'autres langages de programmation que PHP peuvent actuellement être utilisés dans une mesure très limitée avec TYPO3. Alors qu'il est possible, et même courant, pour des langages compilés, de mélanger plusieurs langages de programmation au sein d'un projet, cela reste encore à un stade expérimental pour des langages interprétés. Des extensions expérimentales PHP existent pour intégrer du JAVA et du .NET dans du code PHP. Inversement, il est aussi possible d'accéder à des objets PHP depuis du code JAVA. Cela signifie que des extensions en JAVA sont certainement possibles. Mais elles devront intégrer tous les fichiers PHP nécessaires à une extension. Cela signifie qu'un « wrapper » PHP sera nécessaire pour appeler le code JAVA.

En attendant, il est possible, bien sûr, d'appeler des programmes externes avec `exec()` ou des ressources externes en tant que services Web. Mais il ne s'agit pas d'une intégration complète.

L'intégration devient un thème de plus en plus important partout dans le monde de l'informatique, certainement grâce à l'émergence, ces dernières années, de standards d'interopérabilité. PHP évolue aussi en ce sens. Par conséquent, il est fort probable que les possibilités d'intégration seront encore plus importantes à l'avenir.

7.12 Outils pour le développeur

7.12.1 ExtDevEval

Le module backend **Tools** → **ExtDevEval** fournit un ensemble de fonctions utiles pour le développement d'extensions.

Notez que lors de l'utilisation de l'extension **ExtDevEval** dans le backend, une série de liens apparaissent en haut de l'interface backend. Ceux-ci ouvrent la fenêtre qui contient la documentation de l'API, ou qui renvoie à la documentation sur TYPO3.org.

De plus, le module contient les fonctions suivantes :

getLL() convertir

Convertit les textes au sein du code source pour leur utilisation dans les fichiers localang

PHP script documentation help

Insère les commentaires Javadoc manquants dans les fonctions.

Create/Update Extensions PHP API data

Lit les commentaires JavaDoc et génère, à partir de ceux-ci, un fichier `ext_php_api.dat` qui sert à la création de la documentation de l'API.

Display API from "ext_php_api.dat" file

Affiche la documentation de l'API à partir d'un fichier `ext_php_api.dat`.

temp_CACHED files confirmed removal

Supprime les fichiers `temp_CACHED_*` de `typo3conf/`

PHP source code tuning

Mise en forme du code source PHP et modification pour respecter les conventions d'écriture.

Code highlighting

Affiche PHP, TypoScript et le XML en couleurs. Utile lors de la consultation de la documentation.

CSS analyzer

Présente le CSS pour les éléments au sein d'un code HTML spécifié.

Table Icon Listing

Affiche les possibilités d'icône pour un enregistrement (caché, accès réservé, ...)

7.12.2 Débogage avec debug()

Référence 705356

TYPO3 fournit un peu de support pour le débogage PHP sous la forme d'une fonction globale `debug()`. Cette fonction affiche des variables directement sous un format lisible, y compris pour des tableaux imbriqués.

La fonction `debug()` transfère les variables à afficher, si elles sont disponibles, à la méthode `$GLOBALS['error']->debug()`. Cet objet n'est pas créé par TYPO3 lui-même et peut donc être mis à disposition par une extension. L'extension `CCDebug` (`cc_debug`) en est un exemple. Elle collecte les informations de débogage et les affiche dans sa propre fenêtre, qui peut être activée en cliquant sur l'icône représentant une bombe.

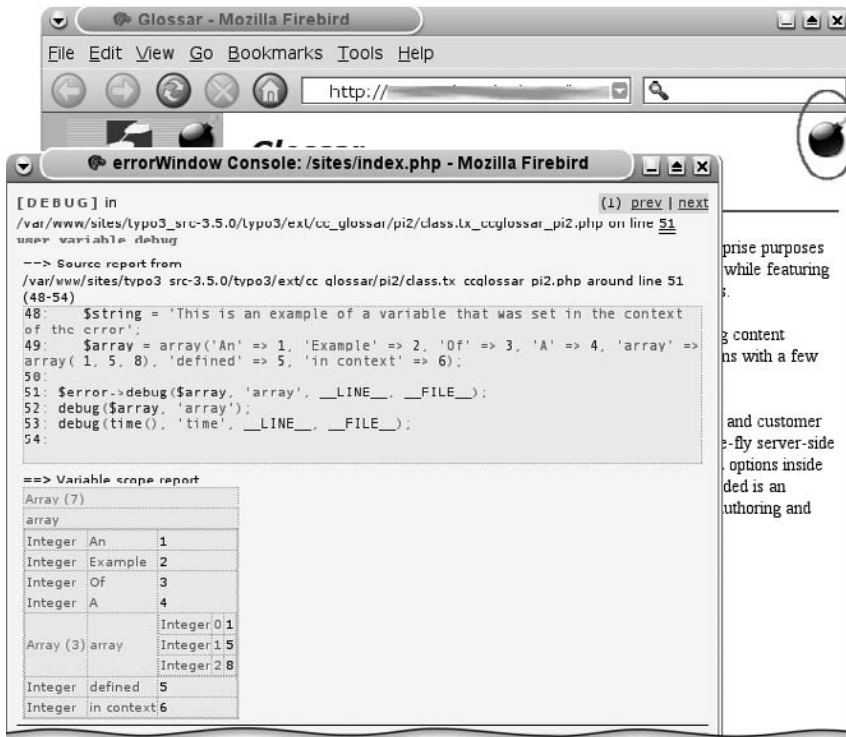


Figure 7.50:
Débogage dans une
fenêtre séparée par
CCDebug

La fonction debug est définie comme suit :

```
function debug($variable, $name = '*variable*', $line = '*line*',
    $file = '*file*', $level = E_DEBUG)
```

La fonction incorporée ne renvoie que la variable elle-même, ainsi que sa description dans \$name. Tous les autres paramètres sont ignorés. Par contre, ils sont pris en compte par CCDe-
bug, qui affiche le numéro de la ligne, et le nom du fichier, pour autant que les constantes
__LINE__ et __FILE__ soient passées en arguments.

```
debug(time(), 'current time', __LINE__, __FILE__);
```

Si vous voulez utiliser la fonction debug incorporée, alors que l'extension de débogage est installée, vous pouvez appeler la fonction xdebug().

Extension Debug

Il est assez facile de développer votre propre extension de débogage. Au sein du fichier ext_lo-
calconf.php, vous devez créer un objet dans \$GLOBALS['error'], qui contient la fonction de-
bug() avec les arguments mentionnés plus haut. Si le but de cette extension est, de manière
similaire à CCDebug, de collecter les informations et uniquement de les afficher dans la page,
on doit aussi y inclure la méthode debugOutput().

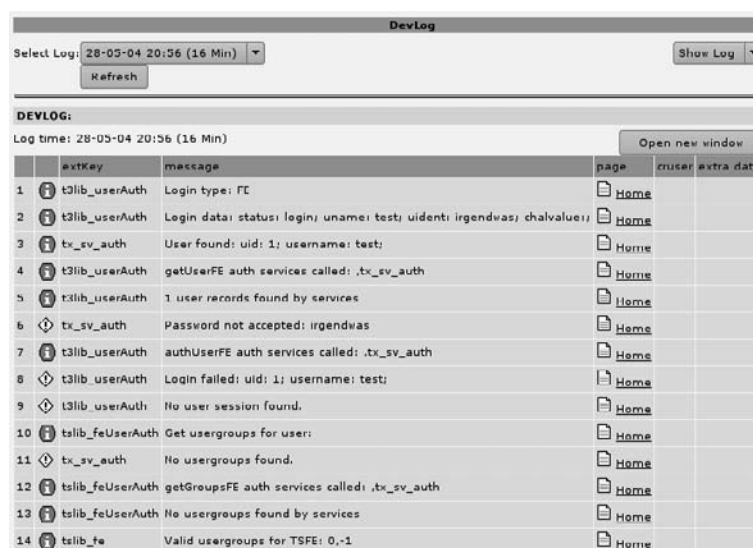
7.12.3 Débogage avec t3lib_div::devLog()

Référence 544345

Outre la possibilité de créer un format d'affichage pour les recherches d'erreur avec la fonction `debug()`, TYPO3 offre une seconde interface via la fonction `t3lib_div::devLog()`. Par opposition à `debug()`, TYPO3 ne met pas en œuvre une fonction par défaut pour `t3lib_div::devLog()`. Cette dernière met à disposition uniquement une interface homogène, de sorte qu'un appel à `t3lib_div::devLog()` sans extension associée n'a pas le moindre effet.

`CCDevLog` (`cc_devlog`) est une extension mettant en œuvre l'interface de (`cc_devlog`) qui enregistre les traces (logs) dans une table de la base de données. Un module backend vous aide à afficher les données. Ce module peut aussi s'ouvrir dans une fenêtre séparée, le rendant disponible parallèlement au backend.

Figure 7.51:
Module `CCDevLog`
pour l'affichage des
logs



The screenshot shows the 'DevLog' interface. At the top, there's a 'Select Log:' dropdown set to '28-03-04 20:56 (16 Min)' and a 'Show Log' button. Below it is a 'Refresh' button. The main section is titled 'DEVLOG:' and shows 'Log time: 28-03-04 20:56 (16 Min)'. There's an 'Open new window' button. The log entries are in a table with columns: 'extKey', 'message', 'page', 'cruser', and 'extra data'. The entries are numbered 1 to 14, each with an icon indicating severity (info, warning, error, or OK).

	extKey	message	page	cruser	extra data
1	t3lib_userAuth	Login type: FC	Home		
2	t3lib_userAuth	Login data: status: login; uname: test; uident: irgendwas; chalvalue:	Home		
3	tx_sv_auth	User found: uid: 1; username: test;	Home		
4	t3lib_userAuth	getUserFE auth services called: ,tx_sv_auth	Home		
5	t3lib_userAuth	1 user records found by services	Home		
6	tx_sv_auth	Password not accepted: irgendwas	Home		
7	t3lib_userAuth	authUserFE auth services called: ,tx_sv_auth	Home		
8	t3lib_userAuth	Login failed: uid: 1; username: test;	Home		
9	t3lib_userAuth	No user session found.	Home		
10	t3lib_feUserAuth	Get usergroups for user:	Home		
11	tx_sv_auth	No usergroups found.	Home		
12	t3lib_feUserAuth	getGroupsFE auth services called: ,tx_sv_auth	Home		
13	t3lib_feUserAuth	No usergroups found by services	Home		
14	t3lib_fe	Valid usergroups for TSFE: 0,-1	Home		

La fonction `devLog()` dans `t3lib_div` est définie comme suit :

```
* @param string Message (in english).
* @param string Extension key (from which extension you are
*               calling the log)
* @param integer Severity: 0 is info, 1 is notice, 2 is warning,
*               3 is fatal error, -1 is "OK" message
* @param array Additional data you want to pass to the logger.
* @return void
*/
function devLog($msg, $extKey, $severity=0, $dataVar=FALSE)
```

Voici un extrait de `t3lib_userAuth`, qui illustre l'utilisation de `t3lib_div::devLog()`.

```
if ($TYPO3_CONF_VARS['SC_OPTIONS']['t3lib/class.t3lib_userauth.php']
    ['writeDevLog']) $this->writeDevLog = TRUE;
if (TYPO3_DLOG) $this->writeDevLog = TRUE;
```

```
...
```

```
if ($this->writeDevLog) t3lib_div::devLog('No user session found.',
't3lib_userAuth', 2);
```

Un test est d'abord effectué pour vérifier si `writeDevLog` est défini dans `$TYPO3_CONF_VARS` au sein du script courant. Si c'est le cas, `$this->writeDevLog` est mis à `TRUE`. La constante `TYPO3_DLOG` est traitée exactement de la même façon. Il est donc possible d'activer les logs uniquement pour ce script ou pour un objet, ou encore de l'activer pour tout le système, grâce à la constante `TYPO3_DLOG`. L'appel lui-même à `devLog()` est toujours précédé d'un test sur la variable `$this->writeDevLog`.

Extension DevLog

Le principe de l'extension de type DevLog est de déclarer une fonction qui sera ensuite appelée par `t3lib_div::devLog()` avec les données de log. Ceci est illustré ci-dessous grâce à un extrait de l'extension `CCDevLog`.

localconf.php

La fonction `devLog()` de la classe `tx_ccdevlog` dans `class.tx_ccdevlog.php` est d'abord déclarée dans le fichier `localconf.php` de l'extension `CCDevLog`.

```
$TYPO3_CONF_VARS['SC_OPTIONS']['t3lib/class.t3lib_div.php']['devLog']
[$_EXTKEY] = 'EXT:'.$_EXTKEY.'/class.tx_ccdevlog.php:tx_ccdevlog
->devLog';
```

Comme vous pouvez le remarquer, votre propre fonction DevLog est définie dans le tableau `'devLog'` avec votre propre clé d'extension : `...['devLog'][$_EXTKEY]`. Cela signifie que plusieurs extensions DevLog peuvent être définies en même temps. Elles sont alors appelées les unes après les autres, avec les données de log.

class.tx_ccdevlog.php

La fonction `devLog()` déclarée plus haut est située dans ce fichier. Dans votre propre extension, ce fichier doit bien sûr être nommé pour correspondre à la clé d'extension. Dans le cas de l'extension `CCDevLog`, les données de log transférées sont traitées et enregistrées dans la table de la base de données.

```
class tx_ccdevlog {
    /**
     * DevLog function - writes log to db
     *
     * @param    array        log data array
     * @return    void
     */
    function devLog($logArr) {
        $insertFields = array();
        $insertFields['msg'] = $logArr['msg'];
```

```

$insertFields['extkey'] = $logArr['extKey'];
$insertFields['severity'] = $logArr['severity'];
if (!empty($logArr['dataVar'])) {
    $insertFields['data_var'] =
        $GLOBALS['TYPO3_DB']->quoteStr(serialize($logArr['dataVar'],
        'tx_ccdevlog');
}
$GLOBALS['TYPO3_DB']->exec_INSERTquery('tx_ccdevlog',
    $insertFields);
}
}

```

Une autre option de débogage, complètement indépendante de TYPO3, consiste à utiliser des environnements de développement tels que Zend IDE ou PHPEclipse. La description de ce processus dépasse le cadre de ce livre. Néanmoins, nous introduirons brièvement quelques IDE.

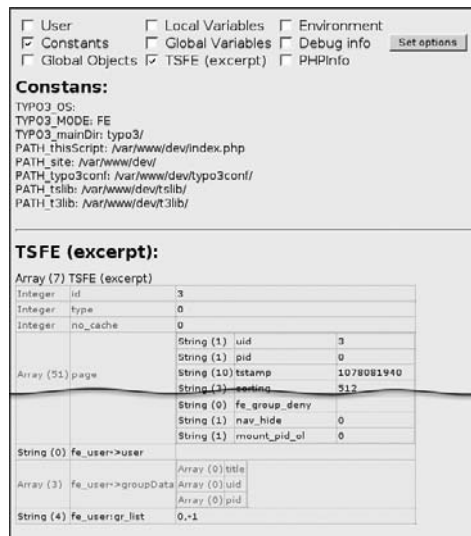
7.12.4 FE Debug/Info output et BE Env-Info

Il est parfois utile d'afficher les données d'objets typiques tels que `$TSFE` et `$BE_USER` ou les variables d'environnement de `t3lib_div::getIndpEnv()`.

On peut consulter ces données grâce, par exemple, aux extensions **FE Debug/Info output** (`cc_feinfo`) et **Backend Environment Information** (`cc_beinfo`), sans devoir générer les résultats via la fonction `debug()`.

La première extension est un plugin que vous pouvez insérer dans une page, la seconde est un module. Le module ne peut pas afficher les données qui sont pertinentes dans le contexte de votre propre module. Mais vous pouvez l'utiliser comme gabarit et facilement générer le résultat dans votre propre module pour vos tests, puisque ces résultats sont intégrés dans une classe séparée.

Figure 7.52:
Exemple d'affichage
du plugin **FE Debug/
Info output**



Si vous voulez intégrer cet affichage pour vos tests dans votre propre plugin, vous le réaliserez très facilement grâce aux lignes suivantes :

```
require_once(t3lib_extMgm::extPath('cc_feinfo') .
    'class.tx_ccfeinfo.php');

$info = t3lib_div::makeInstance('tx_ccfeinfo');
$info->init($this);
$content.= $info->pi_getInfoOutput();
```

7.12.5 Environnements de développement PHP

Si vous programmez de manière intensive, dans des projets complexes, un éditeur de texte n'est pas l'outil le plus adéquat. Mais même un programmeur occasionnel peut bénéficier d'un IDE (*Integrated Development Environment*). Les IDE fournissent généralement une validation de la syntaxe dans l'éditeur, comprennent un système de gestion de projets et intègrent des outils tels que CVS, FTP, et WebDAV. Un bon IDE fournit aussi un support au débogage.

Le choix de l'IDE le plus approprié à vos besoins est une question de goût, raison pour laquelle nous vous recommandons ici d'en tester quelques-uns. Des versions d'évaluation sont généralement mises à disposition par les éditeurs, ou sont même gratuites en tant que logiciels libres.

Voici une courte liste d'environnements de développement PHP :

Environnement de développement Zend

IDE développé par Zend, le concepteur du PHP.
Plate-formes : Unix/Linux, Mac OS X, Windows
<http://www.zend.com>

PHPEclipse

Projet Open Source basé sur Eclipse (<http://www.eclipse.org>), qui met en œuvre un plugin PHP ; l'un des avantages est le grand nombre de plugins dans l'IDE Eclipse : CVS, SQL, XML, HTML, JavaScript, Regex, outils collaboratifs, ...
Plate-formes : Unix/Linux, Mac OS X, Windows
<http://www.phpeclipse.de>

Index

A

- Accès (sous-module) 84
- Access Lists 146
- Accessibilité 353, 356
- Actions 170
- Admin Panel 118
- Administration des utilisateurs 152
- Aide (module) 87
- Aire de navigation 82
- All Configuration 50
- Analyse de la base de données 47
- Analyse du trafic 179
- Arborescence 82
 - des pages 82
 - des répertoires 82
 - vue d'ensemble 84
- Assistant TSConfig 157
- AWStats 178

B

- Backend 79
 - identification 79
- Balise Typo 457
- Basic Configuration 46
- bigDoc 479
- Bogue 74
- Bordure 106
- BPR (Business Process Redesign) 141

C

- Cache 183
- Cadres 346
- Caractères 62
- Caractères spéciaux 56
- Cascading stylesheets 258, 264, 285
- Centre de tâches (sous-module) 86, 87
- Charge élevée 35

- Clé d'extension 369
- Classes de base 407
- Cluster 35
- cms 58
- cObject 229, 438
- Comptes utilisateurs 150
- Conditions 220

- [ELSE] 221
- [END] 220
- browser 340
- dayofmonth 341
- dayofweek 341
- device 340
- globalString 342
- globalVar 342, 344
- hostname 341
- hour 341
- IP 341
- language 341
- loginUser 342
- minute 341
- month 341
- PIDinRootline 342
- PIDupinRootline 342
- system 340
- treeLevel 342
- useragent 341
- userFunc 343
- usergroup 341
- version 340
- conf (tableau) 442
- Configuration (sous-module) 86
- Configuration TS de la page 167
- Configuration TS de la page) 84
- Constant Editor 245
 - cat (sous-catégories) 248
 - clé 247

- commentaires 246
- label 250
- TSConstantEditor (TLO) 250
- type 249
- Constantes 211
- Contenu flexible 362
- Copie recursive 125
- Création de groupes 145
- Cross media (publication) 263
- CSS styled content 258, 284
- CSV export 182
- CType 102
- curl 55

D

- DAM 56, 184
- Database Abstraction Layer 36
- Database Analyzer 47, 74
- Database Mounts 148
- dataMiner 470
- Debogage 55
- Debug 55
- debug() 522
- description de projet 373
- DevLog 525
- diff 59
- Digital Asset Management 184
- Doc (sous-module) 86
- Document Suite 131
- Document TYPO3 252
- Documentation 398
- Droits d'accès 154
- Droits d'accès à une page 154
- Dummy 38

E

- Édition de pages 153

- Édition frontend 155
- Enregistrement de gabarit
 - Include basis template 271
- Environnement de développement 527
- Envoyer des fichiers 116
- Espacement entre cellules 106
- Espacement inter cellules 106
- État d'élément de menu
 - ACT 300
 - ACTIFSUB 300
 - CUR 300
 - IFSUB 300
 - NO 299
 - RO 300
 - SPC 300
 - USERDEF1 300
 - USERDEF2 300
 - USR 300
- Excludefields 147
- exec() 61
- exportation CSV 128
- ExtDevEval 521
- Extension
 - catégories 370
 - clé d'extension 388
 - désinstallation 377
 - DevLog 525
 - Documentation 398
 - documentation 392
 - installation 376
 - Kickstarter 387
- F**
- FEUtilisateur 175
- Fichier (module) 85
- Fichier journal 65, 84, 179
- Fichier T3X 375
- Fichiers (sous-module) 85
- Fichiers, envoi 55
- fileadmin 85
- Filemounts 149
- Flexforms 361, 419
- Fonctions
 - addParams 228
 - encapsLines 228
 - filelink 228
 - HTMLparser 229
 - HTMLparser.tags 229
 - if 228, 334
 - imageLinkWrap 227
 - imgResource 227
 - makelinks 229
 - numRows 227
 - optionSplit 321
 - parseFunc 228
 - select 228
 - split 228, 336
 - stdWrap 227, 282, 291, 331
 - tableStyle 228
 - tags 229
 - textStyle 228
 - typolink 228, 289
- Fonctions (sous-module) 85
- Fonctions de sous-modules 482
- Framework 400
 - structure 400
- Freetype 36
- Frontend 79
 - restitution du contenu 438
- Frontend Editing 117
- Frontend-only 155
- Full Search 181
- G**
- Gabarit TypoScript
 - cascades 271, 281, 288
 - emboîter 205
 - hiérarchie 206
 - mettre en cascade 205
- Gabarit utilisateur 171
- Gabarit-Basis Template 209
- Gabarit-Enregistrement
 - Backend Editor Configuration 211
 - Clear Constants 207
 - Clear Setup 207
 - Constants 207, 211
 - Description 211
 - Include basis template 209
 - Include static 208
 - Include static (from extensions) 209
 - Include static AFTER basedOn 208
 - Resources 207
 - Rootlevel 207
 - Setup 207
 - Static template files from T3 Extensions 209
 - Template on next level 211
 - Template title 206
 - Website title 206
- Gabarits HTML 265
- Gabarits standards 255
 - (example) 264
 - content (default) 257, 279
 - content.tt.* 263
 - cSet (default) 280
 - cSet Stylesheet 281
 - cSet.* 258
 - frameset.* 259
 - language.* 264
 - plugin.* 262
 - styles.* 257
 - styles.content (default) 258, 279
 - temp.* 263
 - template.* 259
- GDLibrary 36
- General Office Displayer 131
- Gestionnaire d'extensions
 - configuration 58
- GFX 50
- GIFBUILDER 305, 315, 325
- GifBuilderObj
 - ADJUST 327
 - BOX 306, 307, 310, 316, 326
 - CROP 327
 - EFFECT 326
 - EMBOSS 326, 329
 - IMAGE 313, 315, 326, 328
 - OUTLINE 326
 - SCALE 327
 - SHADOW 306, 326, 329
 - TEXT 306, 307, 315, 326, 329
 - WORKAREA 326
- GraphicsMagic 36
- H**
- Habillages 507
- Hide in Lists 150
- Historique 132
- Historique des modifications 84
- HMENU/special 318
 - browse 318
 - directory 318, 319
 - keywords 318
 - list 318
 - rootline 318, 320
 - updated 318

userdefined 319
 HTTrack 71

I

Identification 408
 ImageMagick 36
 Images (sous-module) 85
 Importer 85
 Impression de page 179
 Inclusions 221
 Indexation 187
 Indexed Search 32, 111
 info (sous-section) 84
 init.php 478
 Insérer un lien 122
 Insérer un tableau 123
 Insérer une image 122
 Installation
 hardware 32
 LAMP 40
 Linux 39
 Quick install 39
 scenario 33
 test 38
 WAMP 38, 43
 WIIS 44
 Windows 38, 43
 Integrated Development Environment 527
 Internet Information Server 44

J

JavaDoc 522
 Junction 37

K

Kickstarter 387

L

Liste (sous-module) 83
 Localisation (vue d'ensemble) 84
 Logs 65, 178

M

Mémoire tampon 126
 Méta-données 186
 Manuel du rédacteur 78, 87
 Marqueur 273
 Matériel

choix 32
 mediumDoc 479
 Members only 398
 Menu contextuel 504
 Messages (sous-module) 89
 Mises à jour 58, 61, 73
 Module
 framework 478
 script 480
 structure 476
 Module principal 481
 Multilinguisme 344
 multipart/form-data 55
 MySQL 35

N

Navigateur 79
 ActiveX 79
 cache 79
 cookies 79
 RTE 79
 Navigateur d'éléments 96, 116, 185
 Niveau 297
 Niveau racine 167
 No template found 203
 Note rapide (sous-module) 88

O

Objet de contenu (cObject)
 CASE 234
 CLEAR GIF 231, 294
 COA 289, 294, 320, 335, 350
 COBJ_ARRAY 230
 COLUMNS 233, 323
 CONTENT 232, 279, 291
 CTABLE 232, 293
 EDITPANEL 237
 FILE 229, 231, 277
 FORM 235
 HMENU 232, 298
 HRULER 200, 233
 HTML 230
 IMAGE 231, 281, 289, 294
 IMG_RESOURCE 231
 IMGTEXT 233
 LOAD_REGISTER 234
 MULTIMEDIA 237
 OTABLE 233
 PHP_SCRIPT 236

PHP_SCRIPT_EXT 236
 PHP_SCRIPT_INT 236
 RECORDS 232
 RESTORE_REGISTER 235
 SEARCHRESULT 235
 TEMPLATE 236, 286
 TEXT 230, 284, 335
 USER 235, 285
 USER_INT 236

Objet de menu

 GMENU 299, 304
 GMENU_FOLDOUT 299, 311
 GMENU_LAYERS 299, 308
 IMGMENU 299, 314
 JSMENU 299, 317
 TMENU 299, 300
 TMENU_LAYERS 299, 308

Objet de premier niveau (TLO) 237

 config 238, 239
 constants 238, 240
 FEData 238
 includelibs 238
 lib 240
 PAGE 277
 plugin 285
 plugins 238
 resources 238
 sitetitle 238
 styles 239, 240
 temp 239, 240, 287, 338
 TEMPLATE 277
 tt_* 238
 types 238

Office 131

Opérateurs 216

Options générales 94

optionSplit 322

P

page

 types 94

Page (sous-module) 83

Pagemount 61

Pages récentes (sous-module) 88, 129

Panneau d'administration 118, 155

PATH_site 59, 60

Permissions 84

permissions.group 159

PHP_SCRIPT 465

PHP_SCRIPT_EXT 438, 465
PHP_SCRIPT_INT 438, 465
PHPeclipse 527
phpinfo 49
piVars 446
Point de montage fichiers 149
Presse-papiers 83, 126
proxy 55

Q

QuickStart-paquetage 37

R

RAD 143
Rapid Application Development 143
Recherche 181
Ressources 115
Rich Text Editor 86, 119, 162
Rootline 297
RTE 119, 162

S

safe_mode 62, 65
Scénario BT3 270
Service
 clé 512
Services 510
shy extensions 376
Simple hit statistics 84
smallDoc 479
SOBE 481
Sous-groupes 150
Sous-parties 273
SSL 61
Static Templates 208
Statistiques 84, 178
stdWrap (propriétés)
 addParams 334
 case 335
 cObject 332
 current 332
 data 282, 289, 332, 334, 335
 debugData 332
 encapsLines 334
 field 332, 334
 fieldRequired 334
 filelink 334
 filelist 332
 HTMLparser 334

if 334
ifEmpty 333, 337
innerWrap 338
listNum 333, 337, 343
outerWrap 344
override 333, 338
postUserFunc 339
preCObject 338
preUserFunc 339
required 291, 334, 338
split 334, 336
tableStyle 334
textStyle 334
trim 333
typolink 334, 345
wrap 291

Structure des répertoires 402
 t3lib 403
 tslib 404
 typo3 404
SU, changement d'utilisateur 152
Syntax highlighting 245
Système de droits d'accès 142

T

t3d 252
 Export t3d 253
 Import t3d 254
t3lib_BEfunc 479
t3lib_div 409, 480
t3lib_div::devLog() 524
t3lib_extobjbase 496
t3lib_iconworks 480
t3lib_pageTree 497
t3lib_SCbase 479, 486, 496
t3lib_TSpaser 199
Tâches (sous-module) 90
TCE 408
tce_db.php 499, 505
TCEFORM 160
TCEForms 408
TCMAIN 159
template 479
Template Auto-Parser 266, 284
TemplaVoilà 358, 419
 Data Structures (DS) 359
 Template Objects (TO) 359
TER
 Compte utilisateur 395

Test Site (paquetage) 37
Text tools 85, 130
tidy 65
Total Cost of Ownership 34
Trier les pages 125
TS Property Lookup Wizard 157
TSConfig page 159
TSConfig utilisateur 157
TSFE 440
tslib_cObj 441
tslib_content 441
tslib_fe 438
tslib_pibase 445, 446
TSref 199, 282
Type de contenu 102, 103
 formulaire 109
 HTML 115
 identifiant 111
 image 104
 insérer enregistrements 113
 insérer un plugin 114
 lien vers fichier 107
 liste à puces 106
 menu/plan site 112
 multimédia 108
 recherche 111
 séparation 114
 script 114
 tableau 106
 textbox 112, 326
 texte 104
 Texte & image 105
 titre 103
Type de page 95
 avancé 95
 corbeille 99
 création de contenu 101
 délimiteur 98
 dossier système 98
 hors menu 97
 langue 100
 point de montage 98
 raccourci 97
 Rich Text Editor 101
 section utilisateur backend 98
 standard 95
 type de contenu 102
 URL externe 96
 vue en colonnes 99

- typeNum 268
- TYPO3 Source (paquetage) 38
- TYPO3_MODE 520
- TypoScript
 - assistant 242
 - chemin d'objet 214
 - commentaires 215
 - Constant Editor 245
 - constantes 215
 - définition 197, 198
 - gabarits 201
 - objet de contenu 229
 - objets 213
 - opérateurs 214
 - propriétés 213
 - syntaxe 214
 - syntaxe (vérification) 245
- types d'objets 213
- types de données 225
- valeurs 215
- TypoScript Frontend Engine 199
- TypoScript Object Browser 200
- TypoTag 457
- U**
 - unzip 59
 - Usability 77
 - USER 465
 - USER_INT 438, 442, 465
 - Utilisateur (module) 86
 - Utilisateur frontend 175
- V**
 - Vérification de la base de données 180
- Version impression 335, 343
- Vignettes 86
- Visiteurs différents 179
- Voir (sous-module) 83
- Vue étendue 127
- Vue détaillée 82
- W**
 - Web (Module) 83
- X**
 - XCLASS 394, 519
 - xdebug() 523
 - XHTML 353
 - XML Export 182

TYPO3

Développement de sites Web collaboratifs orientés publication de contenu

Outil Open Source et gratuit, TYPO3 fait partie comme SPIP de la famille des CMS (Content Management Systems) basés sur le langage PHP. Plus riche fonctionnellement, mais aussi plus complexe que SPIP, TYPO3 est idéal pour le développement de sites Web ou d'intranets orientés publication de contenu et partage d'informations. Il connaît un succès croissant auprès des administrations, des collectivités locales, des portails d'information ou de presse en ligne, et des entreprises cherchant à mettre en place un intranet de travail collaboratif ou de gestion documentaire.

Le guide de référence des rédacteurs, administrateurs et développeurs TYPO3

Écrit par trois des membres du projet TYPO3, cet ouvrage de référence riche en conseils méthodologiques et en exemples pratiques est découpé en quatre parties :

- Introduction à TYPO3 et installation du produit sous Windows et Linux.
- TYPO3 pour les rédacteurs et éditeurs de contenu, qui découvriront comment insérer et formater des textes et des images, créer des bulletins d'information, des forums de discussion, des agendas d'événements et autres animations éditoriales.
- TYPO3 pour les administrateurs, qui apprendront à gérer les droits d'accès des utilisateurs, à définir les règles de travail collaboratif, à analyser la fréquentation du site et à en optimiser les performances.
- TYPO3 pour les développeurs, qui découvriront comment personnaliser un site en créant de nouveaux gabarits et en développant des fonctionnalités inédites grâce au système d'extensions de TYPO3.

Au sommaire

Installation et prise en main • Les CMS (Content Management Systems) et TYPO3 • Installation de TYPO3 sous Windows ou sous Linux • Configuration du serveur • **TYPO3 pour les rédacteurs** • Interface utilisateur • Modules, centre de tâches • Insertion de contenus • Outils de productivité • **TYPO3 pour les administrateurs** • Gestion des droits d'accès et des utilisateurs • Gestion du workflow • Procédures d'automatisation • Gestion des performances (cache...) • Statistiques et logs • Gestion des fichiers numériques et de leurs métadonnées • **TYPO3 pour les développeurs** • TypoScript : syntaxe du langage, outils de développement, gabarits standards et création de gabarits... • Le système d'extensions de TYPO3 • Développement d'extensions personnalisées.



**W. Altmann,
R. Fritz et
D. Hinderink**

Werner Altmann est un des membres du projet TYPO3 en charge de la documentation officielle. Il est également consultant technique pour une agence berlinoise spécialisée dans les projets CMS d'envergure.

Consultant et développeur freelance, **René Fritz** est un des membres actifs du projet TYPO3. Il a contribué au développement du cœur de TYPO3 et est à l'origine de nombreuses extensions.

Daniel Hinderink est responsable du marketing, de l'innovation et de la stratégie au sein du projet TYPO3. Sa société de conseil basée à Munich assiste les entreprises dans la conduite de projets TYPO3.

Nicolas Wezel a coordonné la traduction de ce livre. En tant qu'expert TYPO3, il donne de nombreuses formations en France et en Belgique. Sa société Streamsys développe des sites Internet et intranet avec TYPO3.